Departamento de Informática

PROYECTO FIN DE CARRERA
Grado en Ingeniería Informática

Diseño, implementación y evaluación de una aplicación de control del tráfico de vehículos

Autor: Jaime Morales Rodríguez de Lope

Tutora: María Soledad Escolar Díaz

Leganés, junio de 2014

Título: Diseño, implementación y evaluación de una aplicación de control del tráfico de vehículos
Autor: Jaime Morales Rodríguez de Lope
Tutora: María Soledad Escolar Díaz

EL TRIBUNAL

Presidente: GARCIA CARBALLEIRA, FELIX

Vocal: SANCHEZ VILLASEÑOR, EDUARDO JESUS

Secretario: ALJUMAILY, HARITH TAHA ABDULLA

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 4 de julio de 2014 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

"Nuestra recompensa se encuentra
en el esfuerzo y no en el resultado.
Un esfuerzo total es una victoria completa."

Mahatma Gandhi

Agradecimientos

A mis padres, Rafa y Mª Ángeles, por su paciencia y empeño para que siguiera adelante después de los pequeños pasos en balde que he dado a lo largo de mi vida.

A mi hermana, Miriam, por su madurez y sensatez a la hora de aconsejarme cuando más lo he necesitado.

A mi novia, Stephany, por su compresión y motivación en los momentos más difíciles.

A mis abuelos, por conseguir lo que ellos nunca pudieron.

Por supuesto, a mi tutora, Soledad Escolar, por saber guiarme a lo largo de todo el proyecto.

Resumen

Este proyecto trata con dos de los conceptos más emergentes actualmente en las Tecnologías de la Información: *Internet of Things* y *Smart Cities.*

Las S*mart Citie*s o ciudades inteligentes permiten responder adecuadamente desde las necesidades más sencillas hasta las más complejas necesidades para proporcionar un incremento de la calidad de vida de los ciudadanos. En particular, este Trabajo Fin de Grado dirige un escenario de *Smart Cities*, que consiste en la representación de una carretera inteligente para el estudio del tráfico de vehículos y cómo su monitorización puede ayudar a mejorar algunos aspectos de la vida cotidiana de los conductores.

La infraestructura de una *Smart City* está generalmente compuesta de multitud de sensores, que permiten detectar diferentes fenómenos para los cuales han sido creados. Los sensores son capaces de transmitir dichos datos en una frecuencia dada. En nuestro caso, cada cierto tiempo se enviarán datos de tráfico de una carretera como intensidad, velocidad y ocupación, entre otros. Estos datos son enviados a un servidor encargado de almacenar y procesar esos datos en tiempo casi real.

Es el servidor el que va a organizar la información en una base de datos, la cual es transparente para el usuario. El servidor realizará consultas sobre los datos en una frecuencia determinada para analizar y permitir mejorar posibles situaciones anómalas en la que se encontraba la carretera.

Las *Smart Cities* se ubican dentro de un concepto mayor que es el de *Internet of Things*, el cual se utiliza para referir a que todas las cosas (*things*) están conectadas entre sí y son capaces de monitorizar su entorno, por lo que pueden proporcionar información del medio en el que se encuentran para posteriormente gestionar los datos y tomar decisiones que contribuyan a mejorar la calidad del entorno de las personas.

Abstract

This project deals with two of the most important concepts in Information and communications technology (IC) at this moment: *Internet of Things* and *Smart Cities.*

The *Smart Cities* are intelligent cities that are able to respond adequately from the simplest to the most difficult needs to increase the quality of life of citizens. In particular, this work addresses a scenario of smart cities which consists in the representation of an intelligent road in order to study the traffic of vehicles and how it may help to improve some aspects of daily life of people.

The infrastructure of a smart city is generally composed of a huge number of sensors that can detect different parameters for which they were created. The sensors are able to transmit these data in a given frequency. In our case, every certain time the sensor will send data of traffic of a road as for instance its intensity, occupation, velocity among others. These data are sent towards a server that stores and processes these data in quasi-real time.

The server is intended to organize the information into a database, which is transparent to the user. The server will also launch queries against the database in a certain frequency in order to detect and be able to enhance some particular scenarios of the roads.

The *Smart Cities* are part of another big concept, the *Internet of things,* which refers to have all possible devices (things) connected together to be able to monitor their environment so they can provide information of the context in which they are. With this information it would be possible to manage data and make decisions to contribute improving the quality of life of people.

Índice

Índice de figuras

Índice de tablas

1. Introducción

Este Trabajo Fin de Grado consiste en la reproducción de una carretera inteligente, la cual va a representar el funcionamiento de una carretera con una serie de sensores integrados que permiten monitorizar datos de control de la misma. El proyecto se enmarca en el ámbito de *Internet of Things*, y dentro de éste en el contexto de *Smart Cities*.

Este capítulo tiene como objetivo proporcionar una introducción a Internet of Things y *Smart Cities* y proporcionar una visión global sobre el área de la ciencia en la que nos adentramos.

1.1. Motivación

Todos podemos tener una imagen de lo que es una ciudad en el futuro. Teléfonos móviles controlando todo tipo de aparatos que usemos habitualmente, como el coche o la puerta de casa; sensores para medir temperaturas y ajustarlas. Vallas de publicidad que cambien dependiendo de la persona que esté pasando o mejoras gracias a los detectores de pérdidas de agua.

A todo esto se le denomina *Internet de las Cosas (IoT)*, que consiste en la conexión a Internet en cualquier momento, en cualquier lugar y de cualquier cosa. Una definición más técnica sería la integración de sensores y dispositivos en objetos cotidianos que quedan conectados a Internet a través de redes fijas o inalámbricas [17]. El hecho de que Internet esté en todas partes es donde aparece uno de los conceptos más importantes de este Trabajo Fin de Grado (TFG): las *Smart Cities.*

Las *Smart Cities*, en castellano conocido como ciudades inteligentes, son ciudades que utilizan la Tecnología de la Información y las Comunicaciones (TIC) para dar a los habitantes facilidades en su vida diaria. Se calcula que en 2050 el 70% de la población mundial vivirá en ciudades; esto nos da una idea muy clara de la importancia que puede llegar a tener dotar de *inteligencia* a las ciudades [1].

De hecho, los avances de las ciudades inteligentes son cada vez más notorios, no sólo por mejorar el conocimiento y el entorno que nos rodea sino también por provocar mejoras en el ámbito medioambiental. Como ejemplo, se puede tener controlado cada segundo, los niveles de contaminación en una zona determinada: cuándo una zona supera unos niveles de contaminación establecida se activan los mecanismos establecidos de corrección. Es por ello que las *Smart Cities* se convierten en un avance con un alto impacto en sectores como la economía, la sociedad, el entorno, el bienestar de las personas.

Con la elaboración de este TFG, se va a conseguir ofrecer uno de los objetivos más importantes de las *Smart Cities* que es la realización de un entorno particular que permita mejorar algún aspecto de nuestras vidas.

En este Trabajo Fin de Grado, se va a implementar una carretera inteligente que va a enviar datos a servidor, el cual va a gestionar los datos que han sido recogidos por sensores. De hecho un usuario que vaya circular por nuestra "mini Smart city" va a saber en casi tiempo-real en qué situación se encuentra la vía por la que va a pasar con su vehículo.

Esta situación permite al usuario decidir si continuar con la ruta habitual o cambiarla para no entrar en el atasco formado. Adicionalmente, podría ver el historial de la carretera para ver a qué hora podría salir de casa para llegar puntual al trabajo.

Por último, esta nueva tecnología abre una puerta con un sinfín de posibilidades que, utilizadas de manera correcta, puede facilitar al usuario su vida cotidiana e indirectamente éste mejora el medio, enviando información que facilitará la vida de los demás.

1.2. Objetivos del proyecto

La finalidad de este proyecto es desarrollar una *Smart Cities* teniendo en cuenta los medios de los que disponemos y ser capaz de mejorar el entorno con la misma. En este caso, es una carretera de una ciudad de la cual se obtienen unos datos con el fin de gestionar la información y mejorar la fluidez.

Los objetivos de este TFG son los siguientes:

- Estudio de los conceptos de IoT y en particular de Smart Cities. Dado el alto impacto que puede tener en el futuro de nuestras ciudades, es necesario revisar la bibliografía relacionada con estos dos conceptos.
- Simulación de una carretera inteligente en el contexto de una Smart City. Dicha carretera inteligente permitirá monitorizar mediante sensores el estado de una carretera. Estos datos sobre el estado serán enviados en tiempo real a un servidor externo el cuál, almacenará, procesará y determinará situaciones anómalas, además de aportar alguna solución para las mismas.
- Análisis, diseño e implementación de una aplicación cliente y servidor que permita implementar la simulación anterior. En particular, se deberá realizar lo siguiente:
    - Creación de sensores (clientes) que detecten información de la carretera y la envíen al servidor.
    - Creación de una base de datos donde se guardará toda la información que haya sido enviada por lo clientes a través del servidor que es quien conecta con ésta.
    - Creación de un protocolo de comunicación que explote las funcionalidades del servidor.
    - Facilitar información a los usuarios del estado en el que se encuentra la carretera.
    - Asegurar la fiabilidad del sistema.

Para conseguir cumplir estos objetivos es necesario el estudio de la tecnología, tanto del hardware como del software y haber hecho correctamente el análisis y diseño del sistema.

1.3. Estructura del documento

Este documento está compuesto por 9 capítulos, una sección de bibliografía y anexos. En el presente capítulo, se pretende ofrecer una visión global del contexto en el que se ha desarrollado el trabajo, la motivación, así como los objetivos del mismo. Al final del capítulo, se muestra una relación de los términos, conceptos básicos y abreviaturas que pueden ser de gran utilidad para una mejor comprensión.

En el capítulo 2, se analiza el *Estado de la cuestión*. Se describirán las tecnologías empleadas para la consecución de este proyecto, así como el hardware y software utilizado en su desarrollo.

En el capítulo 3, se describe el *Modelo del sistema* que es una abstracción del sistema que se va a estudiar, identificando sus principales características y restricciones.

A partir del capítulo 4 se realiza el *Análisis*, donde se afronta el estudio del problema al que se enfrenta el sistema. Más concretamente, se estudiarán los requisitos tanto funcionales como no funcionales del sistema.

El capítulo 5, se centra en el *Diseño* de una solución que cumpla los objetivos planteados al comienzo del documento y los requisitos definidos en el análisis realizado en el capítulo anterior. Se definen de manera detallada también, las decisiones adoptadas para el diseño de la solución planteada, dejando definida tanto la estructura de la aplicación, como la comunicación entre los distintos componentes.

En el capítulo 6, se explica cómo se ha implementado el diseño especificado en el capítulo anterior. Se describirá la *implementación* realizada para cada una de partes programadas.

En el capítulo 7 se realizará la *evaluación* del TFG, donde se demostrarán que los requisitos establecidos anteriormente se cumplen, además de una realización de pruebas de rendimiento y la generación de los resultados.

En el capítulo 8, *conclusiones*, se muestra si se han cumplido los objetivos marcados al comienzo del proyecto. Además, se presentan los gastos inherentes al desarrollo del proyecto, se proponen una serie de mejoras y posibles líneas de trabajo futuras y por supuesto un breve comentario contando lo que ha sido para mí, es decir, mi valoración personal.

En el capítulo 9, *bibliografía*, se muestran los enlaces y documentos de donde se ha obtenido la información.

1.4. Terminología del documento

B

Big data:

En castellano significa “grandes datos”, hace referencia a los sistemas que trabajan con grandes cantidades de datos. Estos sistemas a la hora de almacenar los datos se encuentran con el problema de dónde guardarlos o cómo simplificarlos para que no ocupen tanto espacio.

Bluetooth:

Es una especificación industrial para Redes Inalámbricas de Área Personal (WPANs) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia segura y globalmente libre (2.4 GHz). Los principales objetivos que se pretende conseguir con esta norma son:

- Facilitar las comunicaciones entre equipos móviles y fijos.
- Eliminar cables y conectores entre estos
- Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre nuestros equipos personales.

C

C

Es un lenguaje de programación creado en 1972 por Dennis M. Ritchie en los Laboratorios Bell como evolución del anterior lenguaje B, a su vez basado en BCPL. Al igual que B, es un lenguaje orientado a la implementación de sistemas operativos, concretamente Unix. C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones.

Cisco:

Cisco es el líder mundial en soluciones de redes que transforma el modo en que las personas se conectan, se comunican y colaboran.

CPU

Central Processing Unit (Unidad Central de Proceso), se pronuncia como letras separadas. La CPU es el cerebro del ordenador. A veces es referido simplemente como el procesador o procesador central, la CPU es donde se producen la mayoría de los cálculos. En términos de potencia del ordenador, la CPU es el elemento más importante de un sistema informático.

Crowdsourcing:

Es el método que utiliza Google maps para calcular la velocidad de los vehículos y también de los que están alrededor.

F

FHP

FHP es el Factor de Hora Punta, este término está asociado al capítulo de la Ingeniería del Tráfico. El FHP se calcula mediante el volumen horario y la intensidad de circulación punta dentro de la hora establecida.

G

GPS

El GPS es Sistema de Posicionamiento Global que permite determinar en todo el mundo la posición de un objeto, persona o vehículo con una precisión de hasta centímetros. El GPS a nivel de usuario la precisión son unos pocos metros.

I

IBM

International Business Machines Coorporation (IBM) es una multinacional de tecnología y consultoría que fabrica y distribuye sistemas de hardware y software, adicionalmente ofrece la infraestructura, hospedaje y servicios de consultoría en áreas que van desde ordenadores centrales hasta la nanotecnología [19].

IEE 802.15.4

El propósito del estándar es definir los niveles de red básicos para dar servicio a un tipo específico de red inalámbrica de área personal (WPAN) centrada en la habilitación de comunicación entre dispositivos ubicuos con bajo coste y velocidad (en contraste con esfuerzos más orientados directamente a los usuarios medios, como WiFi). Se enfatiza

el bajo coste de comunicación con nodos cercanos y sin infraestructura o con muy poca, para favorecer aún más el bajo consumo.

IMD

Se conoce dentro del ámbito de la Ingeniería del Tráfico como la Intensidad Media diario anual, más concretamente, es el número de vehículos que pasan por una sección durante un año dividido entre el número total de días del año.

Intel

Intel es el mayor fabricante de circuitos integrados del mundo. La compañía estadounidense es la creadora de la serie de procesadores x86, los procesadores más comúnmente encontrados en la mayoría de las computadoras personales.

El objetivo principal de Intel es diseñar y crear tecnologías que sirven de base para dispositivos informáticos de todo el mundo.

Internet

Conjunto de redes de ordenadores creada a partir de redes de menor tamaño. Es la red global que utiliza el protocolo TCP/IP para proporcionar comunicaciones de ámbito mundial a hogares, negocios, escuelas y gobiernos.

Internet of Things

Internet of Things (IoT) consiste en integración de sensores y dispositivos en objetos cotidianos que quedan conectados a Internet a través de redes fijas o inalámbricas con el objetivo de mejorar la vida de las personas.

IP

Una dirección IP es una etiqueta numérica que identifica, de manera lógica y jerárquica, a una interfaz (elemento de comunicación/conexión) de un dispositivo (habitualmente una computadora) dentro de una red que utilice el protocolo IP (Internet Protocol), que corresponde al nivel de red del Modelo OSI. Dicho número no se ha de confundir con la dirección MAC, que es un identificador de 48 bits para identificar de forma única la tarjeta de red y no depende del protocolo de conexión utilizado ni de la red. La dirección IP puede cambiar muy a menudo por cambios en la red o porque el dispositivo encargado dentro de la red de asignar las direcciones IP decida asignar otra IP (por ejemplo, con el protocolo DHCP). A esta forma de asignación de dirección IP se denomina también dirección IP dinámica (normalmente abreviado como IP dinámica).

IPv4

El Internet Protocol version 4 (IPv4) (en español: Protocolo de Internet versión 4) es la cuarta versión del protocolo Internet Protocol (IP), y la primera en ser implementada a gran escala. IPv4 usa direcciones de 32 bits, limitándola a 2^{32} = 4.294.967.296 direcciones únicas, muchas de las cuales están dedicadas a redes locales (LANs). Por el crecimiento enorme que ha tenido Internet (mucho más de lo que esperaba, cuando se diseñó IPv4), combinado con el hecho de que hay desperdicio de direcciones en muchos casos, se ha visto que escaseaban direcciones, de ahí la aparición de IPv6.

IPv6

IPv6 incrementa el tamaño de la dirección IP de 32 bits a 128 bits para así soportar más niveles en la jerarquía de direccionamiento y un número mucho mayor de nodos direccionales.

El diseño del protocolo agrega múltiples beneficios en seguridad, manejo de calidad de servicio, una mayor capacidad de transmisión y mejora la facilidad de administración entre otras cosas.

Mientras que IPv4 soporta poco menos de 4.3 billones de direcciones, IPv6 ofrece 3.4x10^38(21^28) direcciones IP por cada metro cuadrado de la superficie de la Tierra.

M

M2M

El concepto M2M (máquina a máquina) es, en síntesis, la capacidad de intercambiar datos entre dos máquinas remotas, de forma que mediante este intercambio, es posible controlar y supervisar de forma automática procesos en los que intervienen máquinas. El foco principal de aplicación de M2M se ubica por tanto, en los entornos relacionados con la telemetría y/o el telecontrol.

MySQL

Es un sistema de gestión de bases de datos (SGBD) relacional, multihilo y multiusuario, además es muy conocido y ampliamente usado por su simplicidad y notable rendimiento. Aunque carece de algunas características avanzadas disponibles en otros SGBD.

R

RFID

La identificación por radiofrecuencia (***RFID***, por sus siglas en inglés) es una tecnología de captura e identificación automática de cualquier tipo de información que se halla contenida en pequeños elementos, como etiquetas electrónicas, llamadas en su acepción inglesa "*tags*".

Cuando estos elementos entran en el área de cobertura de un lector RFID, éste envía una señal que será respondida al instante por la etiqueta, transmitiendo la información almacenada en su memoria, normalmente un ***código de identificación***.

Una de las claves de esta tecnología es que la recuperación de la información contenida en la etiqueta se realiza vía radiofrecuencia sin necesidad de que exista contacto físico o visual (línea de vista) entre el dispositivo lector y las etiquetas, aunque sí una cierta proximidad de esos elementos.

S

Sensores

Un sensor es un dispositivo capaz de detectar magnitudes físicas o químicas llamadas variables.

SEVAC

Sistema de Sensores de Variables Atmosféricas en Carretera (SEVAC) está compuesto por distintos sensores de variables atmosféricos, que detectan parámetros que pueden afectar a las condiciones de la circulación, como pueden ser, visibilímetros, anemómetros, pluviómetros, sensores de calzada intrusivos y no intrusivos.

Smart Cities

Las *Smart Cities*, en castellano conocido como ciudad inteligente, son ciudades que utilizan la Tecnología de la Información y las Comunicaciones (TIC) para dar a los habitantes facilidades en su vida diaria, se calcula que en 2050 el 70% de la población mundial vivirá en ciudades por lo que nos da una pequeña idea de la importancia que puede llegar a tener la inteligencia de las ciudades.

Smartphones

Un teléfono inteligente (*smartphone* en inglés) es un teléfono móvil construido sobre una plataforma informática móvil, con una mayor capacidad de almacenar datos, realizar actividades semejantes a una minicomputadora, y con una mayor conectividad que un teléfono móvil convencional. El término «inteligente», que se utiliza con fines

comerciales, hace referencia a la capacidad de usarse como un ordenador de bolsillo, y llega incluso a reemplazar a un ordenador personal en algunos casos.

Sockets

Un socket (enchufe), es un método para la comunicación entre un programa del cliente y un programa del servidor en una red. Un socket se define como el punto final en una conexión. Los sockets se crean y se utilizan con un sistema de peticiones o de llamadas de función a veces llamados interfaz de programación de aplicación de sockets (API, application programming interface). Un socket es también una dirección de Internet, combinando una dirección IP (la dirección numérica única de cuatro partes que identifica a un ordenador particular en Internet) y un número de puerto (el número que identifica una aplicación de Internet particular, como FTP, Gopher, o WWW).

T

TCP/IP

Transmision Control Protocol/Internet Protocol, es el protocolo estandar de comunicaciones en red, utilizado para conectar sistemas informaticos a traves de Internet.

TIC

La Tecnología de la Información y las Comunicaciones (TIC) se desarrollan a partir de los avances científicos producidos en los ámbitos de la informática y las telecomunicaciones. Las TIC son el conjunto de tecnologías que permiten el acceso, producción, tratamiento y comunicación de información presentada en diferentes códigos (texto, imagen, sonido,...).

El elemento más representativo de las nuevas tecnologías es sin duda el ordenador y más específicamente, Internet. Como indican diferentes autores, Internet supone un salto cualitativo de gran magnitud, cambiando y redefiniendo los modos de conocer y relacionarse del hombre.

U

UNIX

El sistema Unix es un sistema operativo que admite múltiples usuarios, así como también múltiples tareas, lo que significa que permite que en un único equipo o multiprocesador se ejecuten simultáneamente varios programas a cargo de uno o varios usuarios. Este sistema cuenta con uno o varios intérpretes de comando (shell) así como

también con un gran número de comandos y muchas utilidades (ensambladores, compiladores para varios idiomas, procesador de textos, correo electrónico, etc.). Además, es altamente transportable, lo que significa que es posible implementar un sistema Unix en casi todas las plataformas de hardware.

Actualmente, los sistemas Unix se afianzaron en entornos profesionales y universitarios gracias a su estabilidad, su gran nivel de seguridad y el cumplimiento de estándares, especialmente en lo que se refiere a redes.

V

VirtualBox

VirtualBox es un software de virtualización para arquitecturas x86/amd64, creado originalmente por la empresa alemana innotek GmbH. Actualmente es desarrollado por Oracle Corporation como parte de su familia de productos de virtualización. Por medio de esta aplicación es posible instalar sistemas operativos adicionales, conocidos como «sistemas invitados», dentro de otro sistema operativo «anfitrión», cada uno con su propio ambiente virtual [21].

VMware

VMware Workstation es un emulador que provee un conjunto de hardware completamente virtualizado al sistema operativo invitado. Además virtualiza todos los dispositivos dentro del entorno virtual, incluyendo el adaptador de video, el adaptador de red, y los adaptadores de disco duro [22].

W

WLAN

Una red de área local inalámbrica, también conocida como WLAN (del inglés wireless local area network), es un sistema de comunicación inalámbrico flexible, muy utilizado como alternativa a las redes de área local cableadas o como extensión de éstas. Usan tecnologías de radio frecuencia que permite mayor movilidad a los usuarios al minimizar las conexiones cableadas.

Wifi

Wi-Fi es un mecanismo de conexión de dispositivos electrónicos de forma inalámbrica. Los dispositivos habilitados con Wi-Fi, tales como: un ordenador personal, una consola de videojuegos, un smartphone o un reproductor de audio digital, pueden conectarse a Internet a través de un punto de acceso de red inalámbrica. Dicho punto de

acceso tiene un alcance de unos 20 metros en interiores y al aire libre una distancia mayor. Pueden cubrir grandes áreas la superposición de múltiples puntos de acceso.

Wi-Fi es una marca de la Wi-Fi Alliance (anteriormente la WECA: Wireless Ethernet Compatibility Alliance), la organización comercial que adopta, prueba y certifica que los equipos cumplen los estándares 802.11 relacionados a redes inalámbricas de área local.

WSN

Wireless Sensor Network o Red de Sensores Inalámbrica, es una red basada en pequeños dispositivos dotados de sensores y capacidad para realizar mediciones, que cooperan entre sí para monitorizar un fenómeno.

Z

Zigbee

Es el nombre de la especificación de un conjunto de protocolos de alto nivel de comunicación inalámbrica para su utilización con radios digitales de bajo consumo, basada en el estándar IEEE 802.15.4 de redes inalámbricas de área personal (Wireless personal area network, WPAN). Su objetivo son las aplicaciones que requieren comunicaciones seguras con baja tasa de envío de datos y maximización de la vida útil de sus baterías.

2. Estado de la cuestión

En este capítulo se explicarán el conjunto de las tecnologías, herramientas y conceptos, que han sido necesarias para el desarrollo de este proyecto.

2.1. *Internet of Things*

Internet of Things (IoT), o Internet de las Cosas, *es un concepto que tiene sus raíces en las redes de Internet, las cuales detectan y procesan la información* [1]. *IoT* consiste en un conjunto de dispositivos interconectados enviándose información. La Figura 1 muestra el funcionamiento del Internet de las Cosas: conectado en cualquier lugar, cualquier momento y a cualquier cosa.

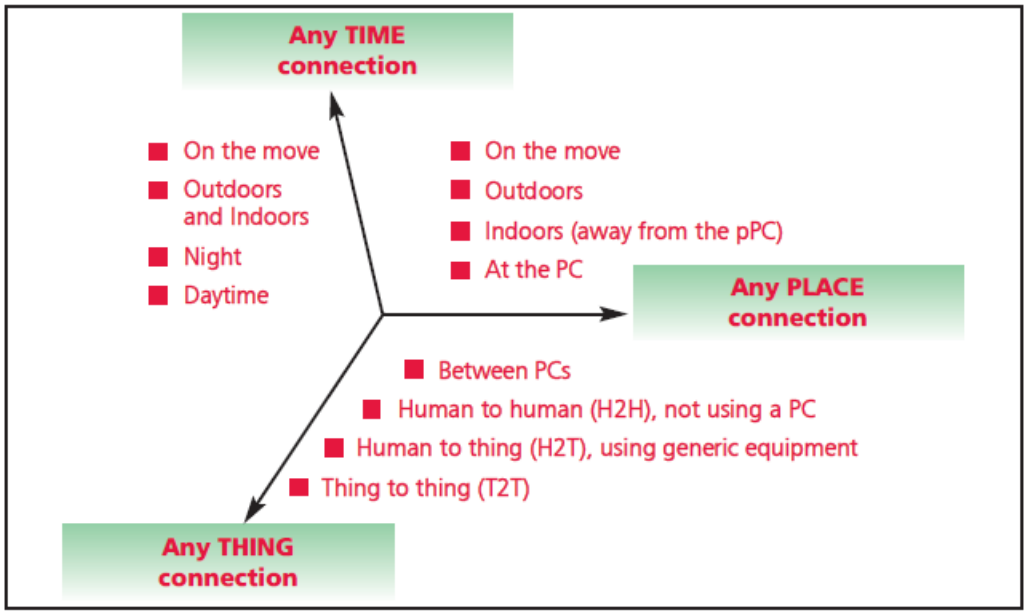

Figura 1: Internet of Things (adopted from Nomura Research Institute)

Este nuevo término va a cambiar la concepción de la vida actual, la educación, comunicación, negocios, ciencia y hasta la humanidad.

Actualmente, se están realizando proyectos de *IoT* en los que se demuestra que mejora todo tipo de análisis, recopilación y distribución de datos sobre todo por su enorme capacidad para ello [2]. Algunos de los proyectos que se están realizando tienen como objetivo evitar las grandes diferencias mundiales que existen entre ricos y pobres, no sólo eso sino en conseguir una mejor distribución de los recursos naturales del planeta.

La aparición de *IoT* se produjo en 1999 en el Instituto de Tecnología de Massachusetts (MIT); apareció gracias al estudio de los sistemas por radiofrecuencia, también conocidos como RFID, y en las nuevas tecnologías de detección de sensores [2].

El gran crecimiento de *IoT* permitirá a cualquier objeto o *cosa* que contenga sensores y funcionalidades de comunicación ser una gran fuente de datos para el resto de la red y es aquí donde aparece el concepto del *Big Data* [3]. El número de objetos conectados a *Internet of Things* está avanzando de manera casi exponencial: en 2003 el número de cosas que estaban conectadas a Internet era de 500 millones y como se puede observar en la Figura 2 en el año 2020 se estima que habrá 50.000 millones de dispositivos conectados. Teniendo en cuenta que la población mundial para esa fecha será de 7.600 millones, se puede afirmar que habrá más de seis dispositivos de media por persona.

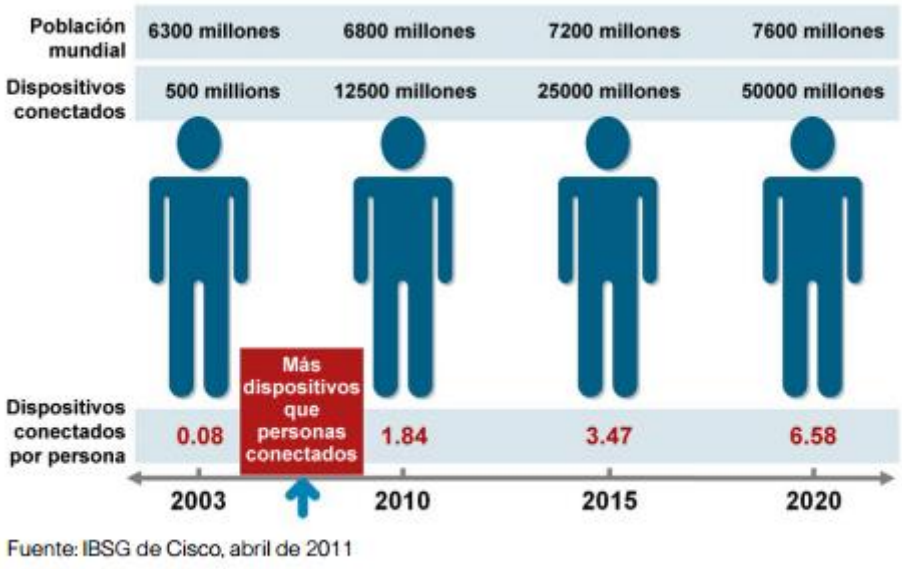

Figura 2: Crecimiento de dispositivos conectados a IoT

La interconexión de los objetos o cosas con Internet representa un gran reto tecnológico, principalmente por el gran volumen y variedad de dispositivos. No sólo estos dos requisitos son importantes sino que también hay que tener en cuenta los requisitos tecnológicos necesarios, como es el bajo consumo o la conexión inalámbrica.

Las redes encargadas de conectar los dispositivos utilizan tecnologías ya existentes, como es Zigbee [4], IEEE 802.15.4 [4] principalmente usados para las WSN. Además, otras de las tecnologías más utilizadas son RFID [4], NFC [4] o Wifi [4]. Bluetooth [4] es otro estándar inalámbrico que permite conectar los dispositivos que forman el IoT, sin embargo está en decadencia debido a su actual velocidad de transferencia y la limitación de radio que posee.

En la actualidad, el IoT se compone de un conjunto disperso de redes dispares diseñadas a medida o *ad-hoc*. El objetivo de esto es que todas las redes se conecten entre sí mejorando las funciones de seguridad, análisis y gestión. En la Figura 3 se puede observar como gracias a IoT todas las redes que antes eran individuales ahora están conectadas.

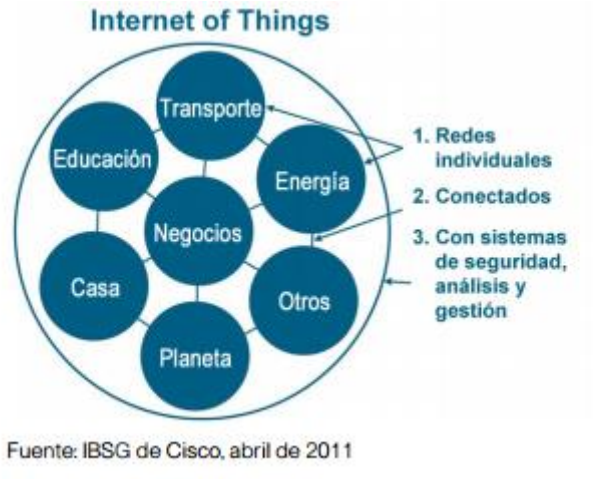

Figura 3: IoT, una red de redes

Por estas razones, se puede afirmar que *IoT* es una evolución de Internet, dado que Internet, la red de redes, puede verse como un conglomerado de redes heterogéneas interconectadas. Los cambios realizados en Internet desde sus orígenes hasta la aparición de *Internet of Things* no han sido tan relevantes como éste último, sino que se limitaban a añadir procesos de desarrollo y mejoras [3].

La aparición de IoT conlleva a que Internet pueda llegar a lugares antes inimaginables, desde aconsejar o desaconsejar un cierto medicamento a través del móvil o hasta a hacer la compra con un simple clic. Por esta razón, la comunicación tanto con el medio como con el resto de las personas es fundamental. De hecho los seres humanos analizamos los datos de manera piramidal, organizados de menor a mayor importancia son: inteligencia, conocimiento, información y datos, tal y como muestra la Figura 4.

Figura 4: Estructura piramidal

Internet of Things se está convirtiendo en algo de vital importancia para las personas, no sólo para mejorar su propia ciudad, entorno o incluso el medio ambiente sino también para proteger su propio planeta. Las personas desean vidas cómodas, saludables y satisfactorias para ellas y para las personas que les rodean. Se necesita además garantizar la sostenibilidad de los recursos (finitos) que consumimos las personas. Gracias a la combinación de IoT con tecnologías de Big Data se puede detectar,

recopilar, transmitir, analizar y distribuir datos a escala masiva con el objetivo de procesar datos que ayuden a la toma de decisiones en tiempo real para garantizar un incremento de la calidad de vida sostenible.

Sin embargo, aún quedan algunos problemas que solventar, entre ellos, el estudio de fuentes alternativas de energía para los sensores que les mantengan operando ininterrumpidamente y su conformidad con los estándares. Es importante destacar la implementación de IPv6 como el protocolo que permitió solventar una limitación de su antecesor IPv4, ya que en 2010 se estaba cerca de agotarse las direcciones IP de IPv4, lo que pudo provocar un freno en el desarrollo de IoT, porque miles de millones de sensores nuevos necesitaban direcciones IP únicas [3]. IPv6 facilita la gestión de redes y añade nuevas características de seguridad como IPSEC. IEEE es sólo una de las organizaciones que se encarga de asegurar que los paquetes con el nuevo protocolo IPv6 puedan enviarse por diferentes tipos de redes. La energía de los sensores también es un factor importante, éstos deberán ser autosuficientes y obtener electricidad a partir del medio físico. Por otro lado, también se debe tener en cuenta los estándares aplicados a los nuevos sensores; se necesita mejorar características como la seguridad, privacidad, arquitectura y comunicaciones. Para finalizar, hay muchas empresas interesadas en la realización de IoT, por el gran potencial que tiene de cara al futuro. Algunas de estas empresas son Cisco, Intel o IBM.

2.1.1. Aplicaciones de IoT

Las posibles aplicaciones de IoT son numerosas y diversas, apareciendo en prácticamente todos los ámbitos de la vida cotidiana de los individuos, las empresas y la sociedad. La Agencia Estratégica de Investigación (SRA) ha identificado y descrito las principales aplicaciones de Internet de las Cosas, que abarcan numerosos ámbitos de aplicación [1].

Las aplicaciones de IoT que reciben más atención son las que están orientadas a las personas pero resultan poco escalables a nivel industrial. La pregunta más lógica es si la implementación de estas aplicaciones se puede extender a sectores más amplios y si es posible redefinir las fases de implementación. Las primeras incursiones de IoT han sido en:

- Smart Cities.
- Smart Environment.
- Smart Water.
- Smart Metering.
- Seguridad y Emergencias.
- Retail.
- Logística.
- Control de Industria.
- Smart Agriculture.
- Smart Animal Farming.
- Domótica y Hogares Automatizados.
- Consumidores

- Sector sanitario.

A continuación se describe cada una de ellas.

2.1.1.1. Smart Cities

Las Smart Cities tienen aplicaciones más específicas, como son las siguientes:

- Aparcamiento inteligente: Monitorización de plazas de aparcamiento disponibles en la ciudad.
- Salud estructural: La monitorización de las vibraciones y las condiciones materiales en los edificios, puentes y monumentos históricos.
- Mapas de ruido urbanos: Sonido de vigilancia en zonas de bar y zonas céntricas en tiempo real.
- La congestión del tráfico: Seguimiento de vehículos y peatones para optimizar los niveles de conducción y rutas de senderismo.
- Iluminación inteligente: Iluminación adaptativa inteligente y tiempo en luces de la calle.
- Gestión de Residuos: La detección de los niveles de basura en los contenedores para optimizar las rutas de recolección de basura.
- Carreteras inteligentes. Carreteras inteligentes con mensajes de advertencia y desvíos de acuerdo a las condiciones climáticas y los eventos inesperados como accidentes o atascos.

Por su importancia para este proyecto, se hablará sobre Smart cities en detalle en la sección 2.1.2.

2.1.1.2. Smart Environment

Internet of Things, también se preocupa por el medio ambiente. Algunos de los dispositivos que ofrecen información pueden llegar salvar, mejorar o incluso rescatar el medioambiente. Entre las aplicaciones destacadas aparecen las siguientes:

Figura 5: IoT Desarrollo Sostenible

- Detector de incendios forestales: Seguimiento de los gases de combustión del fuego de preferencia para definir zonas de alerta.
- Contaminación del aire: Control de las emisiones de CO2 de las fábricas, la contaminación emitida por los automóviles y los gases tóxicos que se generan en las granjas.
- Deslizamiento y prevención de avalanchas de nieve: La monitorización de la humedad del suelo, las vibraciones y la densidad de la tierra para detectar patrones peligrosos.
- Detección de terremotos: Control distribuido en lugares específicos en los que se producen temblores.

2.1.1.3. Smart Water

Algunas de las aplicaciones para *Smart Water* son las siguientes:

- Calidad del agua: Estudio de Análisis del agua en los ríos y el mar para la fauna y la elegibilidad para el uso potable.
- Las fugas de agua: detección de la presencia de líquido y tanques de presión externos variaciones a lo largo de las tuberías.

2.1.1.4. Smart Metering

Las aplicaciones para las mediciones inteligentes son las siguientes:

- Smart Grid: el seguimiento y gestión del consumo de energía.
- Nivel del depósito: control de los niveles de agua, petróleo y gas en los tanques de almacenamiento y cisternas.
- Instalaciones Fotovoltaicas: Seguimiento y optimización del rendimiento en las plantas de energía solar.
- Flujo de agua: Medición de la presión del agua en el transporte de agua sistemas.
- Silos Stock Cálculo: Medición del nivel de vacío y peso de las mercancías.

2.1.1.5. Seguridad y emergencia

IoT también desarrolla aplicaciones para mejorar la seguridad o avisar a agentes de seguridad en caso de emergencia La Figura 6 muestra la relación existente entre la seguridad de los archivos, seguridad de las aplicaciones web y la seguridad en las bases de datos.

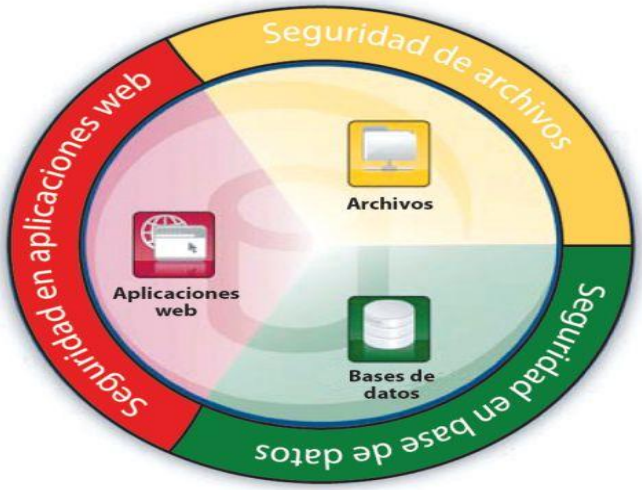

Figura 6: IoT Seguridad y emergencia

Las aplicaciones más destacadas son las siguientes:

- Perímetro de control de acceso: El control de acceso a las áreas restringidas y la detección de personas no autorizadas.
- Presencia de líquido: Detección de líquido en los centros de datos, almacenes y terrenos sensibles de construcción para evitar averías y corrosión.
- Altos niveles de radiación: Medición distribuida de niveles en las centrales nucleares para generar alertas de fuga.
- Gases explosivos y peligrosos: La detección de los niveles de gases y fugas en entornos industriales, alrededores de las fábricas químicas y las minas en el interior.

2.1.1.6. Retail

Las aplicaciones de venta al por menor son las siguientes:

- Control de la cadena de suministro: El seguimiento de las condiciones de almacenamiento a lo largo de la oferta cadena y seguimiento de productos con fines de trazabilidad.
- NFC Pago: Procesamiento de pagos basado en la ubicación o duración de la actividad para el transporte público, gimnasios, parques temáticos, etc.
- Aplicaciones inteligentes comerciales: obtener asesoramiento en el punto de venta de acuerdo a los hábitos de los clientes, preferencias, presencia de componentes alérgicas para ellos o que expiraron las fechas.

- Gestión de productos smart: Control de la rotación de los productos en los estantes y almacenes para automatizar los procesos de repoblación.

2.1.1.7. Logística

Uno de los avances que más facilidades ofrece al ser humano es el hecho de saber con cierto nivel de precisión dónde se encuentra en cada momento un producto o controlar la temperatura constantemente y que se pueda regular, sin necesidad de que una persona tenga que estar vigilando las condiciones en las que se encuentra la producción.

Por esta razón, IoT en cuanto a logística se refiere, puede ser considerado como una revolución industrial. Multinacionales como Microsoft ya se están adaptando lo que provocará que muchas empresas mejoren la logística de su trabajo [13].

Figura 7: IoT Logística

Las aplicaciones más destacadas son las siguientes:

- Ubicación de elementos: Búsqueda de elementos en grandes superficies como almacenes o puertos.
- Almacenamiento, detección, incompatibilidad: Advertencia de las emisiones en los contenedores que almacenan productos inflamables cerrados a otras que contienen material explosivo.
- Rastreo de flotas: Control de las rutas seguidas por los bienes delicados como las drogas médicas, joyas o mercancías peligrosas.
- Calidad de las condiciones de envío: El monitoreo de vibraciones, golpes, la apertura del contenedor o el mantenimiento de la cadena de frío para efectos del seguro.

2.1.1.8. Control de Industria

En cuanto al control de industria, las aplicaciones que existen gracias a IoT son las siguientes:

- Aplicaciones M2M (machine-to-machine): Máquina auto-diagnóstico y el control de los activos.
- Calidad del aire interior: Control de los niveles de gases tóxicos y oxígeno dentro de las plantas químicas para asegurar que los trabajadores y los bienes de la seguridad.
- Presencia de ozono: El monitoreo de los niveles de ozono durante el proceso de secado de la carne en las fábricas de alimentos.
- Control de temperatura: Control de temperatura en el interior industrial y
- refrigeradores médicos con mercancía sensible.
- Ubicación de la cubierta: Activos ubicación interior mediante el uso de activos (ZigBee) y etiquetas pasivas (RFID / NFC).
- Vehículo Auto-diagnóstico: Captura de información para enviar alarmas reales a emergencias o prestar asesoramiento a los conductores.

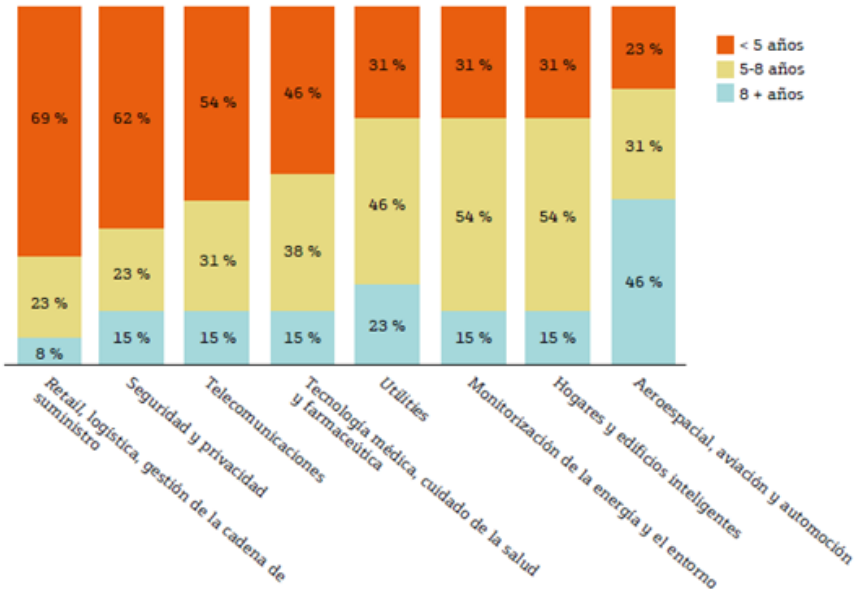

Figura 8: Velocidad de adopción de IoT en las distintas industrias

2.1.1.9. Smart Agriculture

Aunque a priori, IoT va encaminado al estudio de las ciudades inteligentes, también desarrolla aplicaciones para las ciudades más rurales donde se encuentra principalmente el sector primario.

Las principales aplicaciones desarrolladas en cuanto a la agricultura son las siguientes:

- Vino de calidad: Monitorización de la humedad del suelo y el diámetro del tronco en los viñedos para controlar la cantidad de azúcar en las uvas y la salud de la vid.
- *Green Houses*: Control de las condiciones micro-climáticas para maximizar la producción de frutas y hortalizas y su calidad.
- Campos de golf: Riego selectivo en las zonas secas para reducir los recursos hídricos necesarios.
- Estación meteorológica de red: Estudio de las condiciones climáticas en los campos para pronosticar la formación de hielo, la lluvia, la sequía, la nieve o el viento cambia.
- Compost: Control de los niveles de humedad y temperatura en la alfalfa, heno, paja, etc., para prevenir los hongos y otros contaminantes microbianos.

2.1.1.10. Smart Animal Farming

En cuanto a la granja algunas aplicaciones relevantes son las siguientes:

- Hidroponía: Controlar las condiciones exactas de las plantas cultivadas en agua para obtener los cultivos más altos de eficiencia.
- Cuidado de la descendencia: Control de las condiciones de crecimiento de las crías en las granjas de animales para asegurar su supervivencia y la salud.
- Seguimiento animal: Localización e identificación de los animales de pastoreo en pastizales abiertos o la ubicación de los grandes establos.
- Los niveles de gases tóxicos: Estudio de la ventilación y la calidad del aire en las granjas y la detección de gases nocivos de los excrementos.

2.1.1.11. Domótica y Hogares Automatizados

IoT también aparece en los casas de los ciudadanos, mejorando y haciendo más cómoda su rutina. Las aplicaciones son las siguientes:

- Energía y Uso del Agua: Energía y suministro de agua de consumo monitoreo para obtener consejos sobre cómo ahorrar costes y recursos.
- Electrodomésticos de control remoto: Encender y apagar de forma remota los aparatos para evitar accidentes y ahorrar energía.
- Sistemas de detección de intrusos: Detección de aberturas de ventanas, puertas para la prevención de intrusos.
- Arte y Productos Conservación: Monitoreo de las condiciones en el interior de los museos y almacenes de arte.

2.1.1.12. Consumidores

Otra de las aplicaciones que más éxito tiene en IoT son las que llegan a los consumidores, no sólo por la facilidad de compra online con un simple sino también, buscar artículos que en la tienda no están o estar conectado con tu tienda online para recibir la oferta de lo que más te interesa.

Una de las muestras del incremento de estas aplicaciones es el e-commerce, más conocido como comercio electrónico. La facilidad con la que se puede llegar a millones de posibles compradores es tan grande que hasta el pequeño comercio está interesado en este nuevo proyecto.

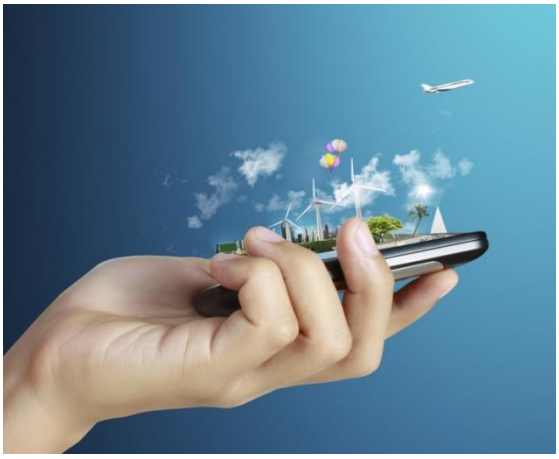

Figura 9: IoT Consumidores

Algunas de las aplicaciones de IoT para los consumidores son las siguientes:

- Control de la cadena de suministro: El monitoreo de las condiciones de almacenamiento a lo largo de la cadena de suministro y seguimiento de productos para fines de trazabilidad.
- NFC pago: Procesamiento de pagos basado en la ubicación o duración de la actividad para el transporte público, gimnasios, parques temáticos, etc.
- Aplicaciones comerciales inteligentes: Obtener consejos en el punto de venta de acuerdo con los hábitos de los clientes, las preferencias, presencia de componentes alérgicas a ellos o fechas de vencimiento.

2.1.1.13. Sector sanitario

Existen sensores en el ámbito de la salud que hacen posible la prevención de enfermedades que pueden llegar a ser mortales. Por ejemplo, se están realizando estudios que identificar el tipo de enfermedad mediante una muestra de la tos de una persona.

Este sector no tiene límites, no sólo por el tema sanitario sino por el valor económico que hay detrás. En Estados Unidos hay veinte millones de diabéticos con los cuales se estima un mercado de ocho mil millones de dólares. Estos estudios consisten en la realización de un glucómetro que permite transmitir los análisis en remoto y recibir

asistencia instantánea on-line. La Figura 10 muestra cómo gracias a IoT está todo conectado, desde el Centro de Salud a los Hospitales Dependientes.

Figura 10: IoT Sector Sanitario

Algunas de las aplicaciones más conocidas en este sector son:

- Detección de caídas: Asistencia a personas mayores o discapacitadas.
- Neveras médicas: Neveras preparadas para medir las condiciones que deben tener los medicamentos, vacunas o elementos orgánicos.
- Cuidado deportivo: Monitorización de los signos vitales en los centros de rendimiento y campos de entrenamiento de deportistas.
- Vigilancia de pacientes: Monitorización de las condiciones de los pacientes dentro de los hospitales y en el hogar.
- Radiación ultravioleta: Medición de los rayos UV del sol para advertir a las personas no ser expuestas en determinadas franja horaria.

2.1.2. Smart cities

Una *smart city o* ciudad inteligente es aquella que utiliza las TIC (Tecnologías de la Información y Comunicaciones) para hacer que tanto su infraestructura, como sus componentes y servicios públicos sean más interactivos, eficientes y que los ciudadanos puedan beneficiarse de ellos [8]. La Figura 11 muestra un ejemplo de *Smart City* con todos los componentes que pueden aparecer en la misma.

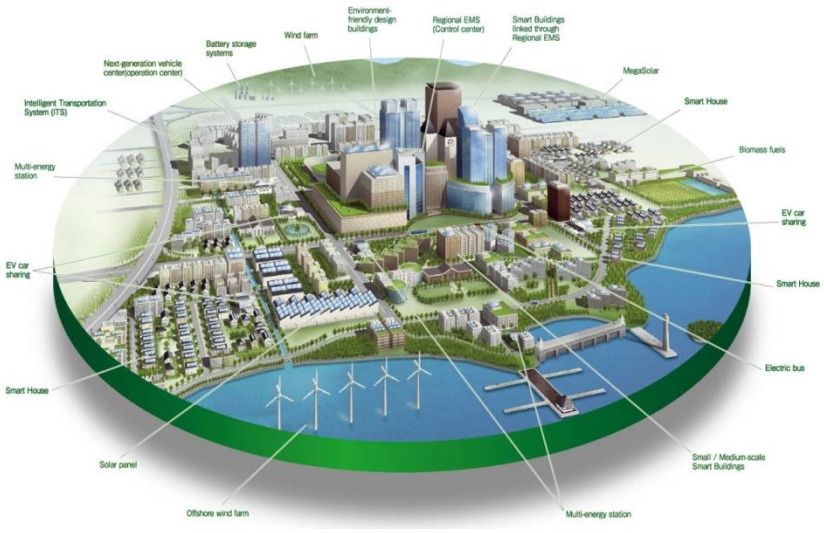

Figura 11: Smart City

Uno de los objetivos principales de las *smart cities* es la mejora de las condiciones para poder beneficiar al habitante de esa ciudad. Una vez conseguido los objetivos de cada red, se unen creando una red de sensores enorme. Esta red está totalmente conectada gracias a las tecnologías que se han comentado anteriormente, como RFID, Wifi, ZigBee, etc.

Existen muchas iniciativas de Smart Cities en el mundo y hay múltiples manera de aproximarse al modelo objetivo final. El servicio que ha dado origen a cada una de ellas es clave para entender su evolución y su idiosincrasia particular [6]. En caso de ciudades inteligentes de nueva creación, se suelen desplegar servicios a la vez que se despliegan infraestructuras, ya que en el análisis y el diseño ya se han tenido en cuenta y adicionalmente se produce una reducción de costes. Este es el caso de la ciudad ***New Songdo*** al oeste de Seúl (Korea) cuyo objetivo es convertirse en un centro internacional de negocios. Se estima un coste en torno a unos 35.000 millones de dólares y albergará a más de 65.000 habitantes [6].

En el caso de ciudades en países desarrollados, la estrategia de la *Smart City* suele incorporarse a la hora de las renovaciones de las infraestructuras de la ciudad, ya que se tiene en cuenta muchas variables como la energética, gestión de recursos, tráfico, agua, etc. De esta manera se consigue que la ciudad sea más eficiente y sostenible. Un ejemplo en España, es la ciudad de Santander, en marcha desde 2009. Se trata de un proyecto [6] de referencia por varios motivos, por un lado por su alcance, ya que está desplegando 20.000 dispositivos en IoT, y por otro lado, se está constituyendo como un gran laboratorio de experiencia en aplicaciones del Internet del Futuro. En el caso de las ciudades de países en vías de desarrollo la aproximación a la *Smart City* suele ayudar a resolver problemas como la congestión del tráfico o densidad urbana, estos problemas surgieron por la rápida urbanización en los últimos 10-20 años. Otro de los objetivos es mejorar el transporte, la comunicación, además de fomentar la industria y el comercio.

Otro de los conceptos relacionados con este tipo de tecnología es el M2M (Machine-to-Machine), *consiste en la utilización de un dispositivo (sensor, medidor, etc.) para capturar un "evento" (temperatura, nivel de inventario, ubicación, estado del medioambiente, etc.) que se transmite a través de una red (inalámbrica, cableada o*

híbrida) y traduce el evento capturado a información significativa (hay una infracción, la máquina expendedora está vacía, el nivel del depósito es bajo, etc.) [7]. La Figura 12 muestra cómo el mundo ofrece información a todo tipo de dispositivos.

Figura 12: Machine to Machine

2.2. Ingeniería del tráfico

Los estudios del tráfico son la herramienta fundamental de la ingeniería aplicada al conocimiento del tráfico para conocer su comportamiento. Esta subsección presenta algunos de los conceptos más relevantes que se necesitan conocer para una gestión adecuada del tráfico de una ciudad [9].

Existen dos tipos de redes viarias, las urbanas y las interurbanas. Las primeras se caracterizan por las frecuentes intersecciones a nivel, pasos de peatones, edificios, etc. donde normalmente la longitud de los desplazamientos de vehículos es corta. En cuanto a las segundas, los vehículos están compuestos por motor y, salvo ocasiones concretas, el tránsito de peatones o vehículos de tracción animal está prohibido. En las redes viarias interurbanas son escasos los terrenos o edificios colindantes y las distancias recorridas son más largas.

Para hacer un correcto estudio del tráfico es necesario tener en cuenta tres variables principales: la *intensidad*, la *densidad* y la *velocidad*. Existen otras variables secundarias que son más esporádicas como por ejemplo, el número de accidentes, los usos y las costumbres de los conductores, tipos de vehículos, centros de ocio cercanos y, por supuesto, las variables meteorológicas.

2.2.1. Factores que afectan al tráfico

Los factores que afectan al tráfico son:

- Intensidad.

- Densidad.
- Factores meteorológicos.
- Relación entre densidad y velocidad.

2.2.1.1. Intensidad del tráfico

La *intensidad* es el número de vehículos que pasan a través de una intersección fija de una carretera por unidad de tiempo. Las unidades más comunes son vehículos/hora y vehículos/día. Normalmente la intensidad del tráfico viene condicionada por la demanda y ésta está condicionada en gran medida por el número de vías.

Al realizar estudios de la intensidad del tráfico en las carreteras se observa que tiene una tendencia creciente a la que se superponen unas oscilaciones cíclicas, que dependen del día, el tiempo, accidentes, etc. El crecimiento de la intensidad es debido al aumento de la población en las ciudades y sobre todo en las grandes áreas metropolitanas, zonas de desarrollo industrial o turismo. Por el contrario, en las zonas rurales donde existe una fuerte emigración o hay épocas de crisis, se puede observar un descenso de la intensidad.

La intensidad se puede clasificar en diferentes ciclos:

- Ciclo anual: La intensidad de un año a otro no suele tener grandes variaciones, teniendo en cuenta ese pequeño incremento de la población en las ciudades. Las variables que influyen en el ciclo anual son las siguientes:
    - El carácter turístico del tráfico.
    - Las bajas intensidades de tráfico que hacen que sean más sensibles a situaciones extraordinarias.
    - La proximidad a una gran población.
    - El carácter industrial de la zona.
    - La mayor proporción de tráfico pesado.
- Ciclo semanal: La intensidad del tráfico difiere de los días laborables (lunes a viernes) de los que no lo son (sábados y domingos). Esta variación se acusa tanto en las vías urbanas como las interurbanas.
- Ciclo diario: Es el más importante desde el punto de vista técnico. La intensidad del tráfico a lo largo del día depende mucho de la hora, los mínimos registran desde las 3 hasta las 5 de la mañana, pero a la entrada de las grandes ciudades desde las 8 a las 9 de la mañana es cuando se anotan los registros las intensidades más altas, a los que se conoce como hora punta. Uno de los factores más importes es el factor de hora punta FPH que se conoce como el análisis de la capacidad de la sección teniendo en cuenta las intensidades, y se calcula:

$$FHP = \frac{\text{Volumen horario}}{\text{Intensidad de circulación punta (dentro de la hora)}}$$

Los datos del ciclo del día dependen mucho de la estación del tiempo en que nos encontremos y son los más sensibles ya que una carretera con una intensidad media diaria baja puede aumentar un día en concreto por un accidente. Otro ejemplo pueden ser las carreteras cercanas a las zonas costeras, donde en verano la intensidad será más alta que un día cualquiera de invierno.

En la composición del tráfico, es interesante diferenciar el tipo de vehículos que pasan por una carretera ya que cada uno de los componentes poseen unas características diferentes y de ahí se pueden obtener datos que diferencien unas zonas de otras.

- Motocicletas:
- Vehículos ligeros
- Vehículos pesados.

Otras definiciones a tener en cuenta en la intensidad son:

- IMD: Se conoce como la intensidad media diaria anual, es el número de vehículos que pasan por una sección durante un año dividido entre 365.
- Intensidad hora punta es el número de vehículos que pasan por una sección durante la hora que se considera representativa de las condiciones de mayor circulación.

2.2.1.2. Densidad del tráfico

La *densidad* es el número de vehículos que hay en un tramo de carretera por unidad de tiempo en un tramo de la carretera. Si el tramo de carretera es un punto fijo se conoce como ocupación. La ocupación se mide con el porcentaje de tiempo que el sensor está ocupado.

La densidad (D) es la variable que va a medir la fluidez del tráfico, ya que al aumentar la densidad, va a disminuir la velocidad (V) y a aumentar la intensidad (I). Si la densidad se acerca a su valor máximo, se circula con paradas y arranques frecuentes. El valor máximo que puede alcanzar la densidad es el máximo número de vehículos que pueden presentarse en una carretera por unidad de longitud. En caso de la ocupación sería el 100%. La siguiente fórmula relaciona la intensidad con la densidad y la velocidad como I=V x D, donde I es la intensidad de circulación (en vehículos/hora), V es la velocidad media de recorrido (en km/h) y D es la densidad (en vehículos/km).

Figura 13: Ejemplo de densidad alta

En la Figura 13, se puede observar un claro ejemplo de densidad alta, suponiendo que los vehículos estén totalmente parados, la velocidad sería 0 y en consecuencia, la intensidad también.

2.2.1.3. Velocidad del tráfico

La *velocidad* se define como la tasa de movimiento expresada como la distancia recorrida por unidad de tiempo, normalmente en km/h. Un concepto de relevancia que influye a las otras dos variables importantes para el estudio de la ingeniería del tráfico es la velocidad de percentil 85, que es aquella que sólo es sobrepasada por el 15% de los vehículos considerando sólo los turismos, dado que son los más rápidos. Esta velocidad suele ser superior en un 20% a la velocidad media y se define como velocidad de proyecto. De hecho, para los estudios de trazado o regulación se considera como velocidad de proyecto puesto que si se considera la velocidad media como velocidad de proyecto sería superada por el 50% de los vehículos. Existen tres tipos de velocidades medias, según el artículo de la DGT sobre la ingeniería del tráfico [9]:

- Velocidad media de recorrido: Se calcula tomando la longitud de un segmento de carretera y dividiéndola entre el tiempo medio de recorrido de los vehículos que la recorren.
- Velocidad media en movimiento: En esta definición no se tiene en cuenta las demoras producidas por las paradas, interrupciones completas o congestión de tráfico.
- Velocidad media temporal: Es la velocidad media de todos los vehículos que atraviesan una determinada sección de la carretera durante un cierto intervalo de tiempo.

2.2.1.4. Factores meteorológicos

Son obtenidas a través de los Sensores de Variables Atmosféricas en Carretera (SEVAC) y miden:

Figura 14: Ejemplo de influencia de los factores meteorológicos

- Precipitación
- Presión atmosférica
- Radiación solar
- Humedad
- Velocidad y dirección del viento
- Visibilidad
- Temperatura

La Figura 14 representa un ejemplo de cómo los factores meteorológicos influyen en las vías tanto urbanas como interurbanas.

2.2.1.5. Relación entre la densidad y la velocidad

Por último, analizaremos la relación entre la densidad y la velocidad. Si la densidad es pequeña quiere decir que hay pocos vehículos en la carretera por lo que se puede circular a una velocidad que podría llegar al máximo de la vía establecido ya que ningún otro vehículo se puede interponer.

Con dificultades mayores es más complicado hacer una afirmación debido a que la densidad puede ser alta pero la velocidad también lo puede. Sí es cierto que la influencia de la densidad se va a notar en el descenso de la velocidad media de la carretera en ese momento.

Por lo que cuando no existen coches sobre la vía, la densidad es cero, la intensidad de la circulación también lo es. En cambio cuando la densidad tiende al máximo todos los vehículos se paran.

La intensidad de circulación de cualquier vía es su capacidad. La densidad de cuando se produce esto se denomina densidad crítica y la velocidad a la que esto ocurre se denomina velocidad crítica.

2.2.2. Detectores de tráfico

En la actualidad existen una gran variedad de dispositivos para recoger datos sobre el estado del tráfico (detectores). La mayoría de ellos son capaces de medir el número de vehículos (intensidad), la velocidad de circulación, el tipo de vehículo y la ocupación de la vía como porcentaje de tiempo. La Figura 15 muestra cómo perciben la información los sensores que se encuentran a lo largo de la vía.

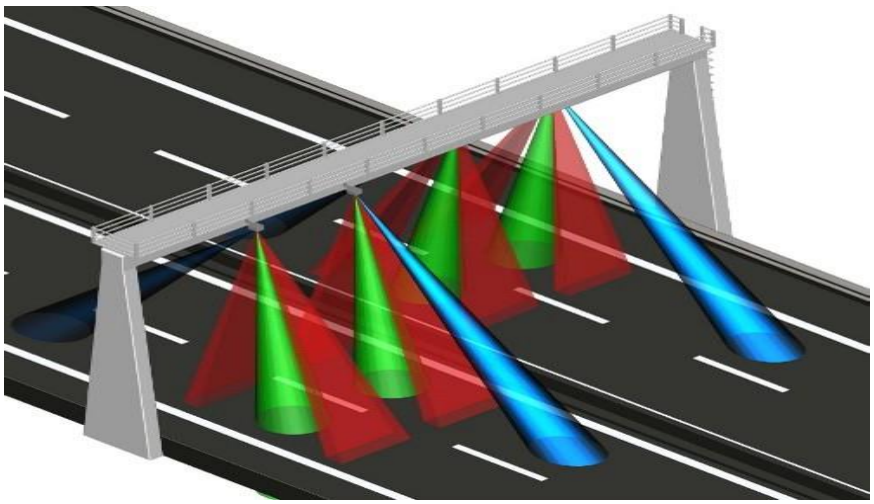

Figura 15: Ejemplo de detector de tráfico

Existen distintos tipos de detectores:

- **De lazo inductivo:** Se basa en el principio de inducción electromagnética. En el pavimento se realizan unos recortes en forma de cuadrados de 2m de lado con una hoja de sierra de dientes diamantados de 7mm, se crea una bobina enterrando el cable en esos cortes y luego se rellena con resina. Cuando un vehículo pasa, al tener masa metálica induce una corriente que es interpretada como el paso de un vehículo, mientras que el tiempo sirve para medir la ocupación de la vía. De esta manera se puede calcular la velocidad a la que pasa un vehículo como V=d/ (t2-t1), donde:
  - V: Velocidad de los vehículos.
  - d: distancia existente entre las dos ruedas o pares de ruedas.
  - t2: Momento exacto que equivale a la segunda vez que pasa la rueda.
  - t1: Momento exacto que equivale a la primera vez que pasa la rueda.

- **De visión artificial**: Se basa en el tratamiento de imágenes capturadas por una cámara de televisión, de forma que se pueden obtener los mismos parámetros que con el detector de tráfico anterior. La gran ventaja de este tipo de detectores es la posibilidad del uso de la imagen en video lento para observar todo tipo de incidencias.

- **De radar de microondas**: Emiten energía a altas frecuencias en el sentido al que se desplazan los vehículos por el cambio de la señal emitida debido al efecto

Doppler, que es proporcional a la velocidad del vehículo. Tienen como ventaja su buen funcionamiento en condiciones meteorológicas adversas y poseen una gran precisión al medir la velocidad. El inconveniente que poseen es que cuando la velocidad baja de 10km/h el dato que ofrece es como si la carretera estuviera vacía. Actualmente se han desarrollado de detectores de verdadera presencia (*true presence*) que eliminan esta desventaja.

- **De infrarrojos**: Se basan en la utilización de un sensor de fotones colocado en un poste o puente junto al carril del que se desea obtener información. Cuando entra un vehículo entra en la zona de detección gracias al calor se produce un cambio en la energía radiada. Este detector únicamente mide la intensidad, además no son muy populares por su escasa exactitud y a que no detectan los vehículos cuando van a bajas velocidades.

- **De ultrasonido**: Emiten ondas de sonido perpendicularmente sobre la carretera, la presencia de un vehículo se determina por la diferencia de tiempos en llegar la onda reflejada, ya que se puede reflejar en un vehículo o en el pavimento. La frecuencia de las ondas emitidas está situada en el rango de 25 a 50 KHz. En cuanto a las ventajas de estos sensores es que son muy fáciles de instalar pero en contra, son detectores muy sensibles a la temperatura y al viento.

- **De captador magnético**: Detectan la distorsión del campo magnético producida por el paso de una masa metálica. Los normales no son capaces de detectar la dirección del movimiento por lo que se mejoran construyendo los llamadas detectores magnéticos compensados, los cuales sí pueden distinguir el sentido de la marcha de la circulación. Este tipo de detectores tienen la ventaja de la fácil sustitución del sensor y que no unos sensores no influyen a otros en caso de proximidad. Pero son muy perturbados por el tendido eléctrico, raíles o tranvías y campo de acción no es muy definido.

- **Técnica basada en Bluetooth**: Las nuevas tecnologías ofrecen una nueva posibilidad de toma de datos a partir de la detección de los dispositivos móviles que usan un protocolo de comunicaciones estándar, el Bluetooth. La antena de detección implantada y receptora de tales señales, analiza y envía en tiempo real la velocidad, hora de paso y sentido del movimiento. Esta tecnología aún está en desarrollo y actualmente hay algunas que están más desarrolladas y pueden ofrecer datos más exactos como los *smartphones* y el GPS.

2.3. Google maps

Google maps [14] es una de las opciones favoritas para escoger un servicio de mapas, ya sea desde el móvil o desde otro dispositivo como portátiles o tablets. Combina diferentes características y funciones que resultan prácticas para los usuarios y aquellos que emprenden un nuevo negocio.

Google maps es uno de los proyectos de Google que más ha evolucionado en estos últimos años, no sólo por su facilidad de uso de cara al usuario sino también por su capacidad de mostrar el tráfico en vivo pudiendo evitar atascos y retenciones tanto en la vías urbanas o interurbanas.

Las operadoras telefónicas conocen exactamente dónde se encuentra el teléfono móvil, entre otras opciones para saber qué tarifas cobrar, por lo que Google se da cuenta de ello y decide crear esta nueva aplicación. Además con la aparición de los *Smartphone* con Android podría ser el único que tuviese esa información y sacar datos en vivo.

Los datos que obtiene Google maps proceden de pequeños reportes que revelan coordenadas del GPS, con esta intención, una combinación de métricas y algoritmos, Google determina la ubicación y la dirección de los usuarios.

El tráfico lo obtiene gracias a la velocidad a la cual se está moviendo el vehículo y el resto de coches que tiene alrededor, este método se conoce como *"crowdsourcing"*. Además Google utiliza mecanismos para diferenciar entre las personas que van andando y los coches de la carretera.

Una vez obtenidos esos datos, que es la fuente principal de información, Google compra información de sensores de tráfico instalados en las vías y adicionalmente tiene un histórico de datos. Con la suma de estos tres factores saca los resultados que muestra en la archiconocida aplicación *Google maps.*

Figura 16: Ejemplo de Google maps

Un ejemplo de captura se muestra en la Figura 16 donde se puede observar el tráfico existente entre las distintas vías de la ciudad. Los colores indican las distintas velocidades a la que van los vehículos, aunque sí es cierto que alguna vez la información no es exacta, *Google maps* intenta mejorar cada día para hacer que su aplicación sea lo más precisa posible.

- Verde: más de 80km por hora.
- Amarillo: 40 - 80km por hora.
- Rojo: menos de 40 km por hora.

- Rojo/Negro: muy lento, arranque y parada de los coches.
- Gris: no hay datos disponibles.

2.4. Lenguaje de programación C

El lenguaje de programación C es también conocido como "Lenguaje de programación de sistemas" desarrollado en 1972 por Dennis Ritchie para UNIX, un sistema operativo multiplataforma [15].

Figura 17: Lenguaje C

Es un lenguaje estructurado de propósito general, aunque en su diseño primó el hecho de que fuera especificado como un lenguaje de programación de sistemas, lo que proporciona una enorme cantidad de potencia y flexibilidad. Tiene sentencias parecidas a la mayoría de los lenguajes de programación, como lo son if, else, for, while, do, etc.

El lenguaje C sigue siendo uno de los más utilizados en la industria del software, así como en institutos tecnológicos, escuela de ingeniería y universidades, de hecho prácticamente todos los fabricantes de sistemas operativos soportan diferentes tipos de compiladores C.

2.4.1. Características de C

Las características que diferencian el lenguaje C frente al resto son las siguientes:

- Una sintaxis para declarar funciones, esta información adicional sirve para que los compiladores detecten más fácilmente los errores causados por argumentos que no coinciden.
- Asignación de estructuras (registros) y enumeraciones.
- Preprocesador más sofisticado.
- Una nueva definición de la biblioteca que acompaña a C. Entre otras funciones se incluyen:
    - Acceso al sistema operativo (por ejemplo: lectura/escritura de archivos).
    - Entrada y salida con formato.
    - Asignación de memoria.

- Manejo de cadenas de caracteres.

Las ventajas principales de C sobre otros lenguajes y hacen que siga siendo uno de los lenguajes más usados y populares en la actualidad. Entre otras se caracteriza por:

- Velocidad de ejecución.
- Existen numerosas bibliotecas de C para programadores profesionales que soportan gran variedad de aplicaciones.
- Se puede acceder a memoria de bajo nivel mediante el uso de punteros.
- Interrupciones al procesador con uniones.
- Lenguaje muy simple que permite programar en múltiples estilos.
- Impide operaciones sin sentido.

Como todos los lenguajes también poseen inconvenientes, los más destacados son los siguientes:

- Recolección de basura nativa, aunque existen librerías que lo hacen.
- Soporte para programación orientada a objetos.
- Funciones anidadas.
- Encapsulación.
- Polimorfismo en tiempo de código en forma de sobre carga.

2.4.2. Ejemplo de C

Un ejemplo típico de programación en lenguaje C, sería el siguiente;

```
# include <stdio.h>  /* necesario para utilizar printf */

main() { /* tipo 'int' de retorno implícito */
        int variable; //Declaración de una variable de tipo entero
        printf ("Hola Mundo\n") ;
        return 0;
}
```

Lo que hace este código es declarar una variable e imprimir la famosa frase “Hola Mundo”.

2.5. Bases de datos MySQL

Es un sistema de gestión de bases de datos (SGBD) relacional, multihilo y multiusuario, además es muy conocido y ampliamente usado por su simplicidad y notable rendimiento. Aunque carece de algunas características avanzadas disponibles en otros SGBD [10].

Figura 18: MySQL

2.5.1. Características de MySQL

En este punto se explicarán las características de este sistema de gestión de base de datos, así como las deficiencias, limitaciones o partes del estándar aún no implementadas.

2.5.1.1. Prestaciones

Las características más importantes que ofrece son las siguientes:

- Está desarrollado en C/C++.
- Se distribuyen ejecutables para gran variedad de plataformas diferentes.
- La API se encuentra disponible en nueve lenguajes de programación.
- Destaca por su velocidad de respuesta.
- Se puede utilizar como cliente-servidor o incrustado en aplicaciones.
- Su administración se basa en usuarios y privilegios.
- Altamente confiable en cuanto a estabilidad se refiere.
- Facilidad de configuración e instalación.
- Soporta gran variedad de Sistemas Operativos

2.5.1.2. Limitaciones

Algunas de las desventajas o limitaciones que ofrece MySQL son las siguientes:

- Admite la declaración de claves ajenas o foráneas en la creación de tablas, internamente no las trata diferente que al resto de los campos.
- Los privilegios para una tabla no se eliminan automáticamente cuando se borra una tabla [11].
- Un gran porcentaje de utilidades no están documentadas.
- No es intuitivo como otros programas (ACCESS) [12].
- No incluye características de objetos como tipos de datos estructurados definidos por el usuario, herencia, etc.

2.5.2. Conexión a servidor

En este apartado se explicará cómo conectarse al servidor desde los diferentes lenguajes de programación que soporta.

Para conectarse con el servidor deberemos asegurarnos que admite conexiones, locales o remotas. También es necesario el nombre de *usuario* y la *contraseña* del mismo, normalmente es *root* y la contraseña *root* o no tiene.

El servidor puede recibir solicitudes a través de *sockets* o *pipes* en forma de ficheros locales a la máquina que se está ejecutando. La conexión a un servidor remoto y un nombre de usuario específico de al menos dos argumentos.

- -h para especificar el nombre del servidor.
- -u para el nombre del usuario.

2.5.3. Ejemplo de MySQL

Un ejemplo de MySQL sería el siguiente.

```
mysql> CREATE TABLE shop (
    -> article INT(4) UNSIGNED ZEROFILL DEFAULT '0000' NOT NULL,
    -> dealer  CHAR(20)                 DEFAULT ''     NOT NULL,
    -> price   DOUBLE(16,2)             DEFAULT '0.00' NOT NULL,
    -> PRIMARY KEY(article, dealer));
mysql> INSERT INTO shop VALUES
    -> (1,'A',3.45),(1,'B',3.99),(2,'A',10.99),(3,'B',1.45),
    -> (3,'C',1.69),(3,'D',1.25),(4,'D',19.95);
```

Figura 19: Ejemplo de MySQL

En la Figura 19 se crea una tabla con atributos y posteriormente se ingresan valores a los atributos creados.

2.6. Sockets

Un socket es un descriptor de un punto final de comunicación (dirección IP y puerto) entre un cliente y un programa servidor conectados mediante una red de comunicaciones. Los sockets se crean y se utilizan como un sistema de peticiones o de llamadas de función.

En cuanto a la abstracción de sockets, representa un extremo de una comunicación bidireccional con una dirección asociada y ofrece una interfaz de acceso a los servicios de red en el nivel de transporte, tanto en el protocolo TCP como UDP. Las propiedades de los sockets TCP son las siguientes [31]:

- Fiabilidad de transmisión.
- Mantenimiento del orden de los datos.
- No duplicado de los datos.
- El “modo disponible” en la comunicación.

2.6.1. Tipos de sockets

Los tipos de sockets que existen son los siguientes:

- Stream (SOCK_STREAM): Este tipo de sockets se usa para las comunicaciones fiables en modo conectado, de dos vías y con tamaño variable de los mensajes de datos. El protocolo de comunicación usado en este tipo de sockets es el TCP. Ejemplos: HTTP, Telnet, FTP, SMTP.
- Datagrama (SOCK_DGRAM): Este tipo de sockets se utilizan para la comunicación en modo no conectado, con envío de datagramas de tamaño limitado. Se basa en el protocolo de comunicación UDP. Ejemplo: DNS.
- Raw (SOCK_RAW): Permite el acceso a protocolos de más bajo nivel como el IP, además no tiene un protocolo de comunicación asociado como ocurre con los dos anteriores.
- SeqPacket (SOCK_SEQPACKET): Tiene las características del SOCK_STREAM pero mantiene el tamaño de los mensajes, es decir, es fijo.

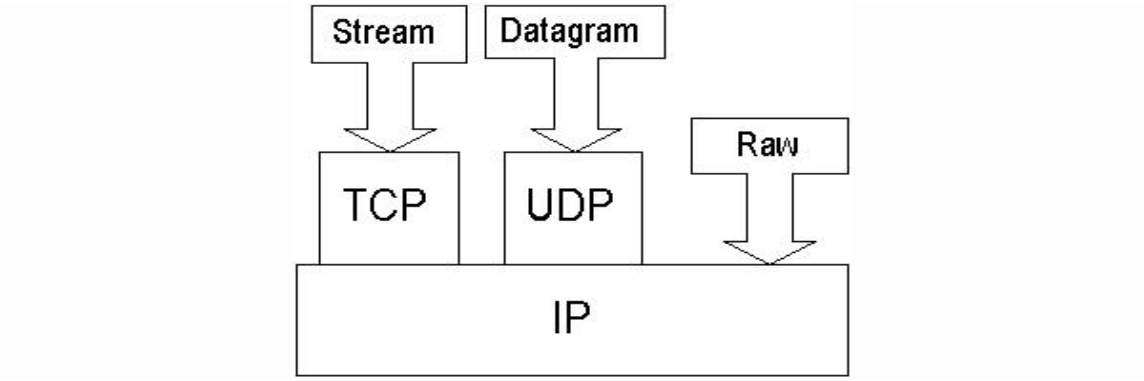

Figura 20: Tipos de sockets

2.6.2. Tipos de dominios

Un dominio representa una familia de protocolos, un socket está asociado a un dominio desde su creación, sólo se puede comunicar con sockets del mismo dominio y los servicios de sockets son independientes del dominio que ofrecen.

Por lo tanto, un dominio representa el formato de las direcciones que podrán tomar los sockets y los protocolos que soportan los mismos. La estructura genérica que poseen los sockets es:

```
struct sockaddr{
```

```
      u_hort sa_famuly; //familia
      char sa_data[10]; //dirección
};
```

Los tipos de dominios que existen son los siguientes:

- Dominio AF_UNIX o AF_LOCAL (Address Family Unix): La comunicación se realiza dentro de una misma máquina, es decir, el servidor y el cliente deben estar ahí. La estructura del dominio es la siguiente:

```
struct sockaddr__un {
      short sun__family;  // en este caso AF_UNIX
      char sun__data[108]; // dirección
      };
```

- Dominio AF_INET (Address Family INET): La comunicación se realiza mediante protocolo TCP/IP, es decir, el cliente y el servidor pueden estar en cualquier máquina. La estructura del dominio es la siguiente:

```
struct in__addr {
      u__long     s__addr;
};
struct sockaddr__in {
      short sin_family;  // en este caso AF_INET
      u__short     sin_port;   /* número del puerto */
      struct in__addr sin__addr; // direcc Internet
      char sin_zero[8];    // campo de 8 ceros
};
```

Existen otros dos tipos de dominios como AF_NS (donde el cliente debe estar en una red XEROX) o AF_CCITT (que se usan para protocolos como CCITT o X25) los cuales no son tan conocidos como los dos anteriores.

2.7. Redes de sensores

Una red de sensores es un sistema distribuido compuesto por un conjunto de nodos con sensores que les permiten muestrear alguna variable de su entorno [32]. Esta clase de redes se caracteriza por su facilidad de despliegue y por ser auto configurables; se tratan de redes ad-hoc, es decir cada uno de sus nodos pueden actuar como emisor, receptor, ofrecer servicios y registrar datos. La Figura 21 presenta la arquitectura de una red de sensores donde un conjunto de nodos sensores conectados inalámbricamente que envían información a un ordenador externo vía un Gateway de comunicaciones.

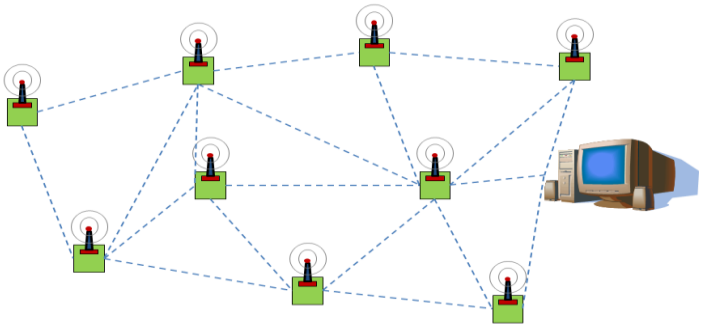

Figura 21: Red de sensores

2.7.1. Características principales de una red de sensores

Las características principales de las redes de sensores son las siguientes, unas son particularidades propias y otras son acogidas de redes Ad-Hoc [31]:

- Topología dinámica: Los sensores deben adaptarse para poder comunicar nuevos datos adquiridos.

- Variabilidad del canal: El canal radio es un canal muy variable ya que existen una serie de fenómenos como pueden ser la atenuación, desvanecimientos rápidos o lentos e interferencias que pueden provocar errores.

- No se utiliza infraestructura de red: Una red de sensores no tiene necesidad de una infraestructura para poder operar, ya que los nodos pueden actuar como emisores, receptores e incluso registrar datos. Sin embargo hay que destacar el hecho de que hay un ordenador conectado a un nodo que va a recoger información y es donde se van a transmitir los datos que se han obtenido por la red de sensores.

- Tolerancia a fallos: Un dispositivo debe tener en cuenta la posibilidad de producir algún error al obtener datos, lo que no debe impedir el funcionamiento del mismo.

- Comunicaciones multisalto o broadcast: En aplicaciones de red de sensores es característico el hecho de usar un protocolo de comunicaciones como los que se explican en el apartado 2.6.1.

- Consumo energético: Es uno de los factores más sensibles. Actualmente no existen energías con capacidad de autonomía ilimitada, lo que provoca uno de los mayores inconvenientes en una red de sensores. Un nodo sensor tiene que contar con un procesador de consumo ultra bajo así como de un receptor radio con la misma característica, lo que hace que el consumo sea aún más restrictivo.

- Limitaciones hardware: Para conseguir un consumo ajustado es totalmente necesario que el hardware sea lo más sencillo posible por lo que se puede afirmar que los sensores tienen una capacidad de proceso limitada.

- Costes de producción: Dada que la naturaleza de una red de sensores tiene que ser en número muy elevada, para poder obtener datos con fiabilidad, los nodos sensores una vez definida su aplicación, son económicos de hacer si son fabricados en grandes cantidades.

  Como ejemplo, la Figura 22 muestra un sensor de los millones que puede llegar a tener una red de sensores.

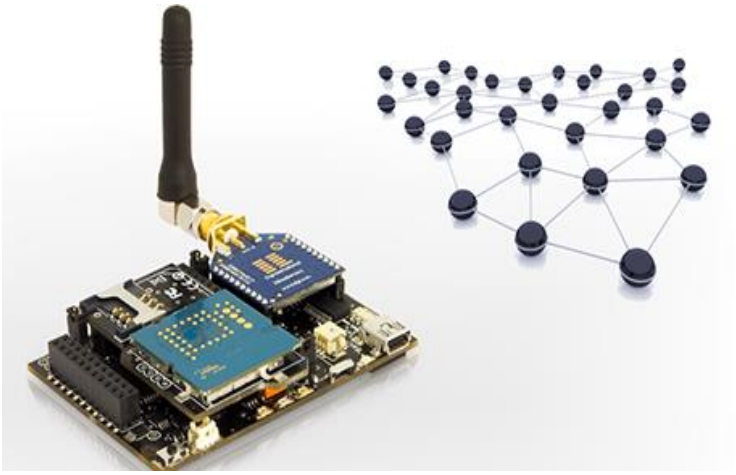

Figura 22: Ejemplo de sensor

3. Modelo del sistema

En este capítulo se explicará el modelo del sistema de la carretera inteligente que se pretende recrear. Un modelo debe representar y describir fielmente el escenario del problema que se pretende resolver, junto con sus características y limitaciones, para poder a continuación desarrollar el sistema que resuelve el problema.

El modelo se basa en un escenario de una Smart City y en particular una carretera inteligente que pretende gestionar el tráfico de la misma. En esta mini-ciudad se pretende modelar un escenario que consiste en una carretera nacional o autopista para gestionar el tráfico de vehículos. A continuación se describen las entidades participantes junto con sus características más relevantes.

Una carretera dispone de un conjunto de carriles (mínimo 1 carril) que conectan dos puntos geográficos (a, b) en ambos sentidos. Para el tramo (a, b) se considera una velocidad máxima de `Vmax`.

Para monitorizar el tráfico de una carretera se dispone de un conjunto de sensores que permiten controlar el tráfico del conjunto de carriles o de manera individual. Estos sensores se pueden dividir en tres tipos, organizados de mayor a menor cantidad de información recogida:

- Sensores de sección: los cuales recogen información del conjunto de los carriles de la carretera.
- Sensores de detector: los cuales recogen información de un carril de la carretera en concreto.
- Espiras: son las que dan la información a los sensores de detector; para comprenderlos, es necesario saber que se colocan dos espiras en paralelo de tal manera que así se pueden poder obtener más datos de los vehículos que pasan.

Sea N el número de sensores de detector de la carretera X, el conjunto de sensores de detector de X viene dado por $D_x = \{D_{x.1}, D_{x,2}, \dots, D_{x,N}\}$. Se asume que las distancias entre los sensores son iguales, es decir, se colocan en puntos equidistantes entre sí.

Por tanto, el número de N de sensores de detector en un tramo de longitud L_my en el mismo carril, siendo la distancia entre detectores d y c el número de carriles que posee la carretera:

$$N_{(a,b)} = \frac{L}{d} * c$$

Cada sensor de detector b_i, siendo ($1 \leq i \leq N$), monitoriza el punto de la vía donde esté situado. Éste incluye los siguientes parámetros:

1) Tiempo de muestreo (t_i). Es el tiempo en la cual se van a realizar las muestras, es decir, la frecuencia.

2) Número de vehículos en un carril, cuya definición exacta es intensidad, se define con I_i. La fórmula para obtenerlo es la siguiente:

$$I_i = \frac{v_i}{t_i}$$

3) Ocupación, se define con el símbolo O_i y es el porcentaje de tiempo que la espira está ocupada.

$$O_i = \frac{\sum(I_i * t_{\text{en cada espira}}) * 100}{t_i}$$

4) Velocidad media en t_i. En cada t_i, el sensor de detector acumula las velocidades de los vehículos que pasan por el mismo.

$$v_i = \frac{\sum(\text{velocidad}_{\text{vehículo}_i})}{I_i}$$

5) Longitud media en t_i, se define con el símbolo L_i.

$$L_i = \frac{\sum \text{Longitudes } (I_i)}{I_i}$$

6) Distancia media en t_i, se define con el símbolo E_i. La distancia media son los metros que hay entre vehículos al pasar por un mismo punto.

$$E_i = \frac{\sum \text{Distancia entre vehículos}}{(I_i - 1)}$$

donde $I_i \geq 1$. La figura 23 muestra la distancia *d* entre dos coches:

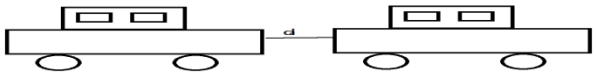

Figura 23: Representación de distancia entre dos coches

7) Número de vehículos pesados en t_i; estos son los vehículos que miden más de 7 metros. Se define como VP_i.

8) Número de vehículos ligeros en t_i; estos son los vehículos que miden menos de 7 metros, se define como VL_i.

$$\text{nº vehículos ligeros} = I_i - VP_i$$

4. Análisis

En este capítulo se va a realizar una exploración objetiva de las especificaciones del proyecto mostrando las particularidades, restricciones y requisitos del problema. Todas estas cuestiones se tendrán en cuenta a lo largo de la construcción del sistema.

4.1. Propósito

El objetivo del proyecto es la construcción de una aplicación distribuida que permita representar una carretera inteligente. En esta aplicación distribuida las tareas se dividen entre los proveedores de recursos o clientes, en nuestro caso los detectores que monitorizan la carretera, y los servidores, que almacenan los datos enviados por los detectores para su posterior procesamiento. En esta ocasión, los clientes (los sensores) enviarán datos al servidor; estos datos pueden ser generados de manera aleatoria o proceder de un fichero de tres días seleccionados ofrecido por la Dirección General de Tráfico (DGT) [18]. Este es el comportamiento típico de una red de sensores donde los sensores generan datos que son almacenados en una estación base. Por su parte, el servidor recibirá los datos, los almacenará en una base de datos y los consultará periódicamente para detectar distintos problemas de tráfico (densidad, distancia, etc.), obtener resultados y tratar de encontrar acciones correctoras para solucionar dichos problemas que pueden surgir como atascos, retenciones o accidentes, además de mejorar la seguridad ofreciendo información sobre la distancia media y la ocupación de la carretera.

Es necesaria una breve explicación sobre los datos generados por los clientes. Como se ha comentado, la DGT nos proporcionó unos ficheros correspondientes a tres días de tráfico. Estos datos nos fueron muy útiles para comprender los datos que se necesitan para controlar el tráfico de vehículos si bien eran limitados ya que correspondía a sólo tres días. Dado que el objetivo de nuestro sistema era la simulación de una carretera inteligente, se decidió que la aplicación cliente pudiera generar además datos aleatorios para poder realizar simulaciones más largas.

4.2. Requisitos funcionales

Un requisito funcional define una función del sistema de software o sus componentes. Por ello, los requisitos funcionales se dividen en cuatro partes fundamentales:

- Requisitos funcionales del servidor.
- Requisitos funcionales del cliente.
- Requisitos funcionales de la base de datos.
- Requisitos funcionales del fichero.

4.2.1. Requisitos funcionales del servidor

Los requisitos funcionales del servidor son los siguientes:

Identificador	**F01**
Nombre	Disponibilidad del servidor.
Descripción	El servidor debe ejecutar un bucle infinito para estar siempre preparado para recibir datos de los clientes.
Validación	Los clientes deben conectarse a través del puerto correcto y la conexión no debe dar error.

Tabla 1: Disponibilidad del servidor

Identificador	**F02**
Nombre	Elegir puerto para realizar la conexión.
Descripción	El servidor debe estar ejecutado a través de uno de los puertos disponibles que son los siguientes: 1024 a 65535.
Validación	Los clientes sólo pueden conectarse a través del puerto elegido en el servidor, sino es así la aplicación da error.

Tabla 2: Puertos que tiene estar activos

Identificador	**F03**
Nombre	El servidor debe ser concurrente.
Descripción	El servidor debe programarse para poder atender a múltiples clientes en un instante de tiempo dado. Creación de un proceso para que cada cliente que se conecte sea independiente al anterior.
Validación	Proteger las condiciones de carrera que se puedan dar por la concurrencia de procesos.

Tabla 3: Servidor concurrente

Identificador	**F04**
Nombre	Almacenar los datos que envían los sensores.
Descripción	Se crea una base de datos donde se guardan los datos enviados por los clientes (sensores).
Validación	Se comprueba que la conexión es correcta y que los datos se han guardado correctamente.

Tabla 4: Conexión con la base de datos

Identificador	F05
Nombre	Calcular y visualizar la velocidad media de los vehículos de la vía en el periodo establecido (regla 1).
Descripción	Acceso a la base de datos para calcular la velocidad media de los vehículos. La velocidad máxima establecida en la vía será la estipulada para cada carretera en particular.
Validación	Comprobación de la media calculada y el mensaje mostrado es correcto.

Tabla 5: Requisito de la regla 1

Identificador	F06
Nombre	Calcular y visualizar la distancia media de los vehículos de la vía en el periodo establecido (regla 3).
Descripción	Acceso a la base de datos para calcular la distancia media de los vehículos. La distancia mínima que debe de haber entre los coches es de 180m.
Validación	Comprobación de la media calculada y el mensaje mostrado es correcto.

Tabla 6: Requisito de la regla 2

Identificador	F07
Nombre	Calcular y visualizar la ocupación media de los vehículos de la vía en el periodo establecido (regla 3).
Descripción	Acceso a la base de datos para calcular la ocupación media de los vehículos. La ocupación se mide en porcentaje, si es menor de 25% la carretera se visualizará en verde, entre 25% y 75% se visualizará en amarillo, y si supera el 75% de ocupación la carretera está en rojo.
Validación	Comprobación de la media calculada y el mensaje mostrado es correcto.

Tabla 7: Requisito de la regla 3

Identificador	F08
Nombre	Ejecución de las reglas de tráfico en la frecuencia establecida y visualización de los resultados.
Descripción	Se crea un proceso que se ejecuta cada cierto tiempo para que acceda a información y calcule las reglas impuestas.
Validación	Se comprueba que el tiempo entre dos ejecuciones consecutivas corresponde a la frecuencia establecida.

Tabla 8: Mostrar resultados reglas en periodo establecido

Identificador	F09
Nombre	Mostrar el estado del servidor.
Descripción	El servidor debe mostrar la situación en la que se encuentra, es decir si está disponible para los clientes o no.
Validación	Si el cliente manda datos que son insertados en la base de datos, el servidor debe decir que está "Esperando conexión".

Tabla 9: Mostrar el estado del servidor

4.2.2. Requisitos funcionales del cliente

Los requisitos funcionales del cliente son los siguientes:

Identificador	F10
Nombre	Elegir puerto correcto.
Descripción	Las conexiones de los clientes deben realizarse a través del puerto escogido en el servidor.
Validación	Los clientes no pueden conectarse con el servidor en un puerto que no esté en el intervalo 1024-65535 y además sino es en el puerto donde el servidor está esperando la conexión.

Tabla 10: Elegir puerto correcto

Identificador	F11
Nombre	Envío de los datos de los clientes al servidor.
Descripción	El cliente envía los datos recogidos de la carretera al servidor. Esto se puede hacer de dos formas: leyendo del fichero proporcionado de la DGT o bien generando datos aleatorios. En caso de error se notifica al cliente que no se han recibido los datos.
Validación	Se comprueba que no da error y en el caso de darlo, sale del programa.

Tabla 11: Conexión del cliente con servidor

Identificador	F12
Nombre	Generar datos aleatorios para simulaciones más largas.
Descripción	En el caso de ejecución del cliente con el parámetro –r 1, el cliente debe generar un fichero aleatorio donde cada línea se compone de los datos cuyo formato es el que espera el servidor. Los datos son enviados al servidor. Por ejemplo: ./cliente –s localhost –p 1200 –r 1
Validación	Se comprueba en la base de datos que se han introducido los datos correctamente.

Tabla 12: Cliente puede generar datos aleatorios

Identificador	F13
Nombre	Cliente puede enviar al servidor los datos leídos del fichero de la DGT.
Descripción	El cliente lee los datos del fichero proporcionado por la DGT. Para cada línea del fichero se envía una serie de datos al servidor. En este caso, a la hora de ejcutar el cliente es necesario especificar la opción –r 0. Por ejemplo: ./cliente –s localhost –p 1200 –r 0.
Validación	Comprobar que se selecciona bien el origen de los datos.

Tabla 13: Cliente puede leer fichero de la DGT

Identificador	F14
Nombre	El sensor deberá enviar datos de la carretera.
Descripción	Los datos enviados por el sensor son los siguientes: • Id. de carretera. • Detector. • Sentido. • Intensidad. • Ocupación. • Velocidad. • Longitud media de los vehículos. • Distancia media entre vehículos. • Agrupación de los vehículos con longitud menor a 7 metros. • Agrupación de los vehículos con longitud mayor a 7 metros. • Agrupación de los vehículos con velocidad menor que 50. • Agrupación de los vehículos con velocidad mayor que 50 y menor que 100. • Agrupación de los vehículos con velocidad mayor que 100.
Validación	Comprobar que se envía cada columna.

Tabla 14: Datos que envíe el sensor

4.2.3. Requisitos funcionales de la base de datos

Los requisitos funcionales de la base de datos son los siguientes:

Identificador	F15
Nombre	Creación de la base de datos.
Descripción	Se crea una base de datos con un nombre y contraseña. El servidor deberá poder conectar a la base de datos a través de dicha información.
Validación	Se comprueba que el servidor se conecta sin errores.

Tabla 15: Creación de la base de datos

Identificador	F16
Nombre	Organizar la base de datos con los atributos que obtienen los sensores, (ver F14).
Descripción	Se crea la tabla con los atributos necesarios para el funcionamiento del servidor.
Validación	Se comprueba que están todos los atributos creados correctamente.

Tabla 16: Creación de la tabla

4.2.4. Requisitos funcionales del fichero

Los requisitos funcionales del fichero son los siguientes:

Identificador	F17
Nombre	Los sensores deben leer los datos del fichero de la DGT especificado como entrada.
Descripción	Se abre para lectura el fichero para extraer datos del mismo.
Validación	Se comprueba que no da error al abrir el fichero.

Tabla 17: Funcionamiento fichero entrada

Identificador	F18
Nombre	El servidor debe mostrar los datos a los usuarios correctamente. En el fichero se mostrarán los datos que están relacionados con las tres reglas, ver F05, F06, F07.
Descripción	Se abre un fichero de log para escribir los datos extraídos.
Validación	Se comprueba que no da error al abrir el fichero y que el mensaje se ha escrito correctamente.

Tabla 18: Funcionamiento de los ficheros de salida

Identificador	F19
Nombre	Ubicación de los ficheros de entrada y salida.
Descripción	El fichero se debe encontrar en la misma carpeta donde se encuentra el ejecutable de la aplicación.
Validación	Comprobar que está el fichero en la misma carpeta.

Tabla 19: Ubicación del fichero

Identificador	F20
Nombre	Atributos entre “;”.
Descripción	Todos los atributos del ficheros deben estar entre “;”.
Validación	Comprobar que entre cada atributo hay “;”.

Tabla 20: Atributos entre ";"

4.3. Requisitos no funcionales

Los requerimientos no funcionales son los que especifican criterios para evaluar la operación de un servicio de tecnología de información, en contraste con los requerimientos funcionales que especifican los comportamientos específicos.

Los requisitos no funcionales del sistema son los siguientes:

Identificador	**F21**
Nombre	SSOO Linux tanto en cliente como servidor.
Descripción	El computador donde ejecutan el cliente y el servidor debe tener instalado un sistema operativo Linux. Se permite la existencia de máquinas virtuales instaladas con el S.O Linux.
Validación	Comprobar Linux en la CPU.

Tabla 21: SSOO Linux

Identificador	**F22**
Nombre	Lenguaje de programación C en servidor.
Descripción	El servidor está programado en C.
Validación	Comprobar que con añadiendo otro lenguaje en el código no funcionaría.

Tabla 22: Lenguaje de programación C servidor

Identificador	**F23**
Nombre	Lenguaje de programación C en cliente.
Descripción	El cliente está programado en C.
Validación	Comprobar que con añadiendo otro lenguaje en el código no funcionaría.

Tabla 23: Lenguaje de programación C en cliente

Identificador	**F24**
Nombre	Lenguaje de programación MySQL para la base de datos.
Descripción	La comunicación con la base de datos es en MySQL.
Validación	Sin conocer el lenguaje no se puede conectar a la base de datos.

Tabla 24: Lenguaje de programación MySQL

Identificador	**F25**
Nombre	Uso de sockets
Descripción	La aplicación cliente-servidor deberá utilizar sockets para la comunicación entre ellos.
Validación	Se comprueba que no da error y en el caso contrario, sale del programa.

Tabla 25: Uso de sockets

5. Diseño

En este capítulo se realizará el diseño del sistema teniendo en cuenta el análisis realizado en el capítulo anterior y se establecerán las bases para la implementación del proyecto.

5.1. Arquitectura del sistema

La arquitectura de nuestro sistema es una arquitectura distribuida que sigue el modelo cliente-servidor. Asumiendo el rol del cliente, tendremos los sensores de la carretera y asumiendo el rol del servidor tendremos una aplicación que hace las veces de servidor concurrente y que permite almacenar y procesar la información enviada por los clientes. La aplicación cliente tiene dos formas de uso:

- Menor escala. Los datos que generan los sensores son datos reales que provienen de ficheros proporcionados por la DGT.
- Mayor escala. Los datos que generan los sensores son datos aleatorios que la aplicación cliente genera. Cada sensor actúa como un cliente que genera y envía los datos al servidor para que éste gestione posteriormente la información recibida.

Sea cual sea la forma de generación de datos de los clientes, el servidor se encargará de recibir los datos, almacenar la información en una base de datos y realizar consultas periódicas con el objeto de extraer información del estado actual del tráfico de la carretera En la Figura 24 se muestra la arquitectura que posee el sistema. Como se puede ver, va a existir un conjunto de clientes. Cada cliente, es un sensor por lo que en cada momento habrá un conjunto distinto de clientes, desde 1 a n. Para poder atender a múltiples clientes generando y enviando datos al mismo tiempo es necesario dotar de concurrencia al servidor de manera que se pueda evitar la pérdida de datos.

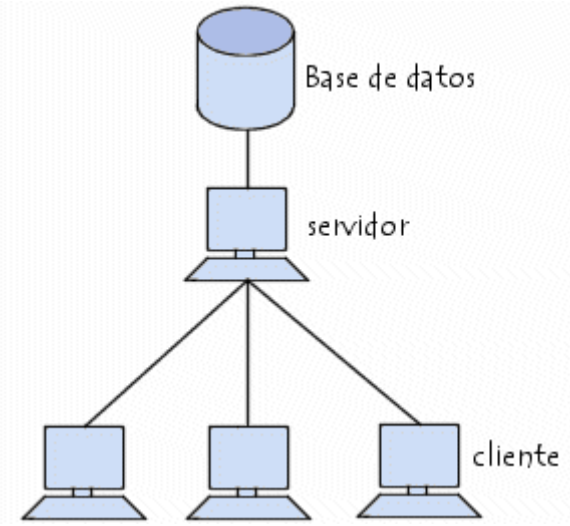

Figura 24: Arquitectura de la aplicación

5.1.1. Clientes (Sensores)

Un sensor es un dispositivo capaz de detectar magnitudes físicas o químicas llamadas variables. Para el desarrollo del proyecto se va a asumir que cliente y sensor son sinónimos, por lo tanto son los clientes los que van a enviar las variables obtenidas al servidor. Como se describió en el capítulo 3, *Modelo del sistema*, existen tres tipos de sensores.

- Sensores de sección.
- Sensores de detector.
- Espiras.

A continuación se describe cada uno de ellos.

5.1.1.1. Sensores de sección

Los sensores de sección son los que reciben información del conjunto de los carriles de la carretera. Éstos pueden enviar la información al servidor de manera independiente de cada carril o lo pueden hacer de manera conjunta obteniendo la media del conjunto de los carriles.

En la Figura 25, se puede observar un ejemplo de sensor de detector el cual obtiene información de la intensidad de la vía de todos los carriles, de tal manera que puede obtener datos para posteriores avisos a los vehículos.

Figura 25: Sensor de detector

5.1.1.2. Sensores de detector

Los sensores de detector son los que recogen información de un carril en concreto, en el caso de que la vía tenga tres carriles existirían tres sensores de detector. En ocasiones las vías interurbanas sólo poseen un carril, por lo tanto el sensor de sección y sensor de detector serían sinónimos.

5.1.1.3. Espiras

Las espiras son las que dan la información a los sensores de detector en la mayoría de las ocasiones, depende del tipo de sensor (véase Capítulo 2.2.5 Detectores de tráfico). Se basa en la colocación de dos espiras, las cuales al pasar los vehículos por encima son capaces de obtener las variables necesarias para realizar el estudio de la vía.

Como se puede observar en la Figura 26, se colocan dos espiras en paralelo para poder obtener las variables a estudiar.

Figura 26: Espiras (x2)

Por tanto, las variables obtenidas por los clientes (sensores) son:

- Intensidad en periodo de carretera (vehículo/periodo): La intensidad es la cantidad de vehículos que pasan por un sensor en un determinado periodo de tiempo.
- Ocupación (%): Porcentaje de tiempo que la espira está ocupada.
- Velocidad media (km/h): Velocidad media de los vehículos que pasan por las espiras.
- Longitud media (dm): Longitud media los vehículos que pasan por esa espira, lo que da información de qué tipo de vehículo es, pesado o ligero.
- Distancia media (m): Distancia media entre los vehículos que pasan por la espira.
- Agrupación por longitud (<=7m): Conjunto de vehículos cuya longitud es inferior o igual a 7 metros.
- Agrupación por longitud (>7m): Conjunto de vehículos cuya longitud es superior a 7 metros.
- Agrupación por velocidad (<50 km/h): Conjunto de vehículos cuya velocidad no supera los 50 km/h.
- Agrupación por velocidad (50-100 km/h): Conjunto de vehículos cuya velocidad se encuentra entre 50 y 100 km/h.
- Agrupación por velocidad (>100 km/h): Conjunto de vehículos cuya velocidad supera los 100 km/h.

Por otro lado cada espira debe estar caracterizada por: carretera, detectores, sentido y fecha

- Carretera: Es el nombre de la carretera donde se encuentra.
- Detectores: Es posición de la espira donde se encuentra.
- Sentido: Es el sentido de la circulación.
- Fecha: El momento en el que se recoge el resultado.

5.1.2. Servidor

El servidor es el que posee la aplicación de gestión de los datos de los sensores y el que realizar la inserción de los datos en la base de datos, y su análisis para la detección y corrección de situaciones en la carretera. Deberá programarse en lenguaje C y deberá ser concurrente, es decir, podrá atender simultáneamente a varios clientes a la vez, como se puede ver en la Figura 27.

Figura 27: Servidor concurrente

El servidor va a almacenar la información que envían los clientes a través de sus conexiones vía sockets. Cuando un cliente envía datos al servidor, éste deberá crear un proceso ligero o hilo que permita atender a cada cliente, de manera que pueden gestionarse múltiples clientes que pueden enviar datos simultáneamente..

Una de las características fundamentales que debe tener el servidor es su disponibilidad; para ello deberá ejecutar en un bucle infinito a la espera de las peticiones de los clientes. Adicionalmente, el servidor posee otro proceso ligero o hilo independiente que se ejecuta cada cierto periodo de tiempo, el cual va a ejecutar tres consultas sobre la base de datos con el objetivo de calcular una serie de información que permita la toma de decisiones sobre la carretera.

La Figura 28 muestra el esquema de comunicación basado en sockets entre un cliente y un servidor.

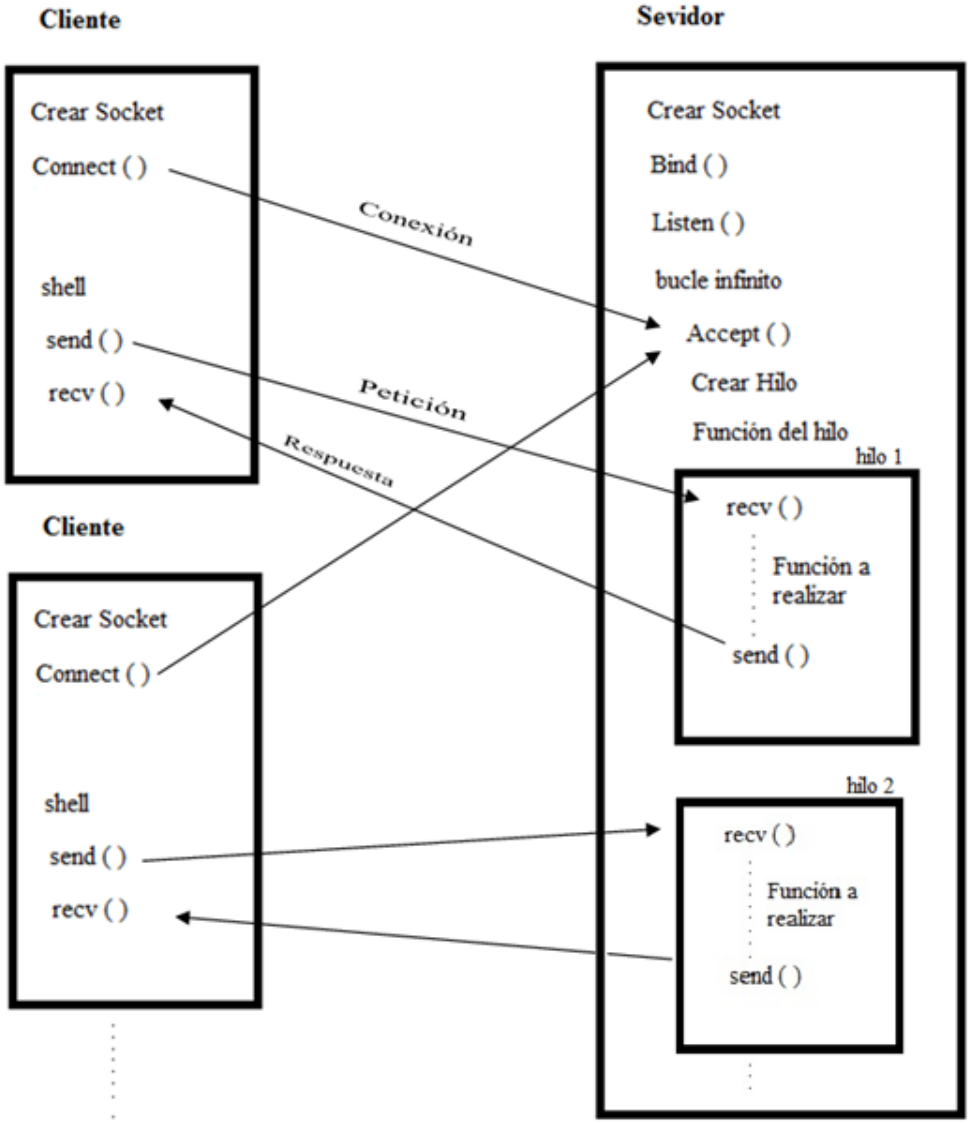

Figura 28: Modelo cliente-servidor

Las funciones de la Figura 28 están explicadas en el capítulo 6.2.

5.2. Modos de operación del cliente

Como se vio al comienzo del capítulo, el cliente tiene dos modos de operación que se explican a continuación:

- Menor escala: usa el fichero de la DGT.
- Mayor escala: genera datos aleatoriamente.

5.2.1. Fichero de la DGT

La Dirección General de Tráfico (comúnmente conocido como DGT) es un organismo autónomo dependiente del Ministerio del Interior de España responsable de la ejecución de la política vial [29].

Un fichero de la DGT almacena los valores recogidos por los detectores en un determinado punto kilométrico de una carretera. Fue la DGT quién nos ha proporcionado tres ficheros distintos correspondientes a tres días diferentes. Estos días han sido elegidos por el autor del proyecto, por lo que uno corresponde a un día festivo, otro a un día lluvioso y otro a un día soleado. Por otro lado el detector dado se corresponde a la misma posición en todos los ficheros, lo que facilita las comparaciones entre un día y otro.

Figura 29: Dirección General de Tráfico

Cada fichero de la DGT consta de catorce columnas, organizadas de la siguiente forma. La Figura 30 muestra las primeras 5 columnas, la Figura 31 muestra las siguientes 5 columnas del fichero y la Figura 32 muestra las últimas cuatro columnas, hasta completar 14.

Carretera	Detectores	Sentido	Fecha	Intensidad en periodo de agreg (Veh/periodo)
A-6	DET12 PK011+400D T01 A-6	DEC	01/11/2013 00:01	7
A-6	DET12 PK011+400D T01 A-6	DEC	01/11/2013 00:02	5
A-6	DET12 PK011+400D T01 A-6	DEC	01/11/2013 00:03	3
A-6	DET12 PK011+400D T01 A-6	DEC	01/11/2013 00:04	4

Figura 30: Fichero DGT Parte 1

Ocupación (%)	Velocidad (km/h.)	Longitud Media (dm)	Distancia Media (m)	Agrupación por longitud (<=7m)
2	100	38	124	7
1	94	41	179	5
1	106	42	196	3
2	93	61	255	3

Figura 31: Fichero DGT Parte 2

Agrupación por longitud (>7m)	Agrupación por velocidad (<50 km/h.)	Agrupación por velocidad (50-100 km/h.)	Agrupación por velocidad (>100 km/h.)
0	0	3	4
0	0	4	1
0	0	1	2
1	0	4	0

Figura 32: Fichero DGT Parte 3

La descripción de las columnas es la siguiente:

- Carretera: Los datos son de la carretera A-6.

Carretera
A-6

Figura 33: Ejemplo Fichero DGT Carretera

- Detectores: Indican el número de detector que corresponde, el kilómetro y los metros donde se encuentra y el carril correcto de derecha a izquierda.

Detectores
DET12 PK011+400D T01 A-6

Figura 34: Ejemplo Fichero DGT Detectores

  - DET 12: Indica que es el detector 12.
  - PK011+400D: Indica que está en el kilómetro 11 más 400 metros.
  - T01: Indica que es el carril de la derecha.

- Sentido: Indica hacia dónde se dirige la circulación.

Sentido
DEC

Figura 35: Ejemplo Fichero DGT Sentido

- Fecha: Indica el día, mes año y minuto, ese minuto completo es el tiempo que ha estado recibiendo datos.

Fecha
11/11/2013 00:01

Figura 36: Ejemplo Fichero DGT Fecha

- Intensidad en periodo: Indica el número de vehículos que han pasado en el minuto que es el intervalo de tiempo elegido.

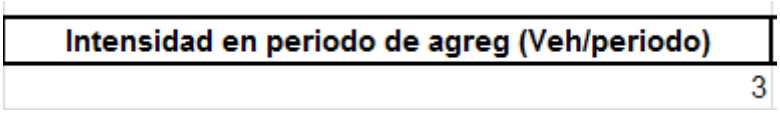

Figura 37: Ejemplo Fichero DGT Intensidad

- Ocupación: Indica el porcentaje de tiempo que la espira está ocupada.

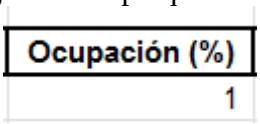

Figura 38: Ejemplo Fichero DGT Ocupación

- Velocidad (km/h): Velocidad media de los vehículos que han pasado en el intervalo de tiempo establecido.

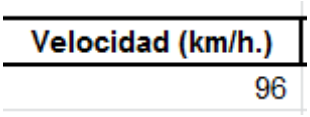

Figura 39: Ejemplo Fichero DGT Velocidad

- Longitud media (dm): Longitud media de todos los vehículos que han pasado en el intervalo de tiempo establecido.

Figura 40: Ejemplo Fichero DGT Longitud media

- Distancia media (m): Distancia media de todos los vehículos que han pasado en el intervalo de tiempo establecido.

Figura 41: Ejemplo Fichero DGT Distancia media

- Agrupación por longitud (<=7m): Conjunto de vehículos cuya longitud es inferior o igual a 7 metros.

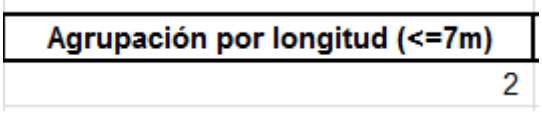

Figura 42: Ejemplo Fichero DGT Agrupación <=7m

- Agrupación por longitud (>7m): Conjunto de vehículos cuya longitud es superior a 7 metros.

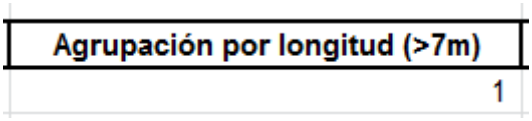

Figura 43: Ejemplo Fichero DGT Agrupación >7m

- Agrupación por velocidad (<50 km/h): Conjunto de vehículos cuya velocidad no supera los 50 km/h.

Figura 44: Ejemplo Fichero DGT Agrupación <50km/h

- Agrupación por velocidad (50-100 km/h): Conjunto de vehículos cuya velocidad se encuentra entre 50 y 100 km/h.

Figura 45: Ejemplo Fichero DGT Agrupación 50-100 km/h

- Agrupación por velocidad (>100 km/h): Conjunto de vehículos cuya velocidad supera los 100 km/h.

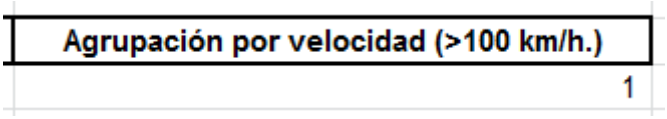

Figura 46: Ejemplo Fichero DGT Agrupación > 100km/h

5.2.2. Fichero Aleatorio

Dado que el número de datos de los ficheros es limitado, para poder permitir simulaciones más largas de la carretera se permite la posibilidad de generar los datos explicados anteriormente de manera aleatoria. Para ello, el cliente va a generar los mismos datos que el fichero de la DGT exceptuando los cuatro primero, que son carretera, detector, sentido y fecha. Como se puede observar en la Figura 47, así es como aparece el fichero aleatorio:

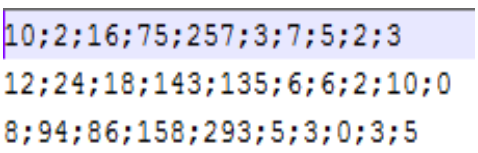

Figura 47: Ejemplo fichero aleatorio

Los datos organizados de izquierda a derecha son los siguientes:

- Intensidad: El fichero aleatorio genera gracias a la función de lenguaje C `srand(rdtsc())` un número el cual tendrá influencia en el resto de las variables. Corresponde al primer número de cada línea.
- Ocupación: En esta variable también se genera un número aleatorio pero si la intensidad es 0, la ocupación también lo es. Corresponde al segundo número de cada línea.
- Velocidad: Ídem. Corresponde al tercer número de cada línea.
- Longitud media: Ídem. Corresponde al cuarto número de cada línea.
- Distancia media: Ídem. Corresponde al quinto número de cada línea.
- Agrupación por longitud menor o igual que 7: Ídem. Corresponde al sexto número de cada línea.
- Agrupación por longitud mayor que 7: Ídem. Corresponde al séptimo número de cada línea.
- Agrupación por velocidad menor que 50: Ídem. Corresponde al octavo número de cada línea.
- Agrupación por velocidad entre 50 y 100: Ídem. Corresponde al noveno número de cada línea.
- Agrupación por velocidad mayor que 100: Ídem. Corresponde al décimo número de cada línea.

5.3. Protocolo de servicio

Una aplicación cliente-servidor debe definir un protocolo de servicio entre las entidades participantes, de manera que puedan ser capaces de intercambiar un conjunto de mensajes entendibles para cada uno. En el servidor es donde se centralizan los diversos recursos y la aplicación que se va a desarrollar en este TFG [28] y el cliente (sensor) es el que se va a encargar de obtener la información de la carretera.

El cliente o sensor enviará los mensajes con una estructura diseñada a priori y a través del protocolo de comunicación TCP mediante sockets (el siguiente apartado justifica el uso de TCP frente a UDP). Esto significa que cada cliente que quiera comunicar con el servidor deberá establecer primero una conexión con el servidor antes de proceder al envío de los mensajes. El protocolo de servicio entre cliente y servidor está formado por dos únicos mensajes:

- Mensaje enviado del cliente al servidor con los datos de los sensores.
- Mensaje de respuesta enviado del servidor al cliente indicando si la recepción y almacenamiento en la BBDD se ha realizado correctamente o no.

5.3.1. Protocolo de comunicación

Para poder implementar la aplicación cliente-servidor es necesario tener en cuenta el protocolo de comunicación que se va a utilizar, por ello en este apartado se va a realizar una comparación de los posibles protocolos a utilizar y el más adecuado para la aplicación.

5.3.1.1. Protocolo de transporte TCP

TCP es un protocolo del nivel de transporte orientado a conexión, donde los datos no se transfieren en registros o bloques sino como *streams* o flujos de datos. Si se rompe la conexión entre los dos procesos comunicantes, éstos serán informados de tal suceso para que tomen las medidas oportunas [30].

El protocolo de comunicaciones con *streams* es un protocolo orientado a conexión, ya que para establecer una comunicación utilizando el protocolo TCP, hay que establecer en primer lugar una conexión entre un par de sockets. Mientras uno de los sockets atiende peticiones de conexión (servidor), el otro solicita una conexión (cliente). Una vez que los dos sockets estén conectados, se pueden utilizar para transmitir datos en ambas direcciones.

5.3.1.2. Protocolo de transporte UDP

Son un servicio de transporte sin conexión. Son más eficientes que TCP, pero en su utilización no está garantizada la fiabilidad. Los datos se envían y reciben en paquetes, cuya entrega no está garantizada. Los paquetes pueden ser duplicados, perdidos o llegar en un orden diferente al que se envió.

El protocolo de comunicaciones con datagramas es un protocolo sin conexión, es decir, cada vez que se envíen datagramas es necesario enviar el descriptor del socket local y la dirección del socket que debe recibir el datagrama. Como se puede ver, hay que enviar datos adicionales cada vez que se realice una comunicación, aunque tiene la ventaja de que se pueden indicar direcciones globales y el mismo mensaje llegará a muchas máquinas a la vez.

5.3.1.3. Diferencias entre TCP y UDP

Las ventajas que ofrece TCP sobre UDP son las siguientes:

- Garantiza tres cosas: que sus datos lleguen, que lleguen en orden y sobre todo que lleguen sin duplicidades.
- Tiene un buen rendimiento relativo a través de un módem o una red de área local.
- Está más indicado para la implementación de servicios de red como un control remoto y transmisión de ficheros.
- Está orientado a conexión, lo único que hay que establecer la comunicación entre dos sockets antes de nada.
- No tiene límite en el tamaño de los datagramas, todos los datos se van leyendo según van llegando.

Por los motivos anteriores, se ha seleccionado TCP en lugar de UDP como protocolo de transporte entre el cliente y el servidor.

5.3.2. Estructura de los mensajes

La estructura del mensaje que va a ser enviado del cliente al servidor es la siguiente:

```
struct sensor{
      char carretera [3];
      char detector [25];
      char sentido [3];
      char fecha [18];
      int intensidad;
      int ocupacion;
      int velocidad;
      int longmedia;
      int distmedia;
      int agrupm7; //Longitud menos que 7 metros
      int agrupM7; //Longitud mayor que 7 metros
      int velm50; //Velocidad menor que 50
      int velm100; //Velocidad menor que 100
      int velM100; //Velocidad mayor que 100

};
```

El mensaje que envía el servidor al cliente simplemente es un entero avisando al cliente si se han insertado correctamente los datos o no.

5.3.3. Intercambio de mensajes

El sensor (cliente) lee o genera los datos de la vía y los almacena en la estructura sensor cuya estructura se presenta en el apartado anterior. Realizará lo mismo para todos y cada uno de los campos que contiene la estructura. Una vez se ha rellenado la estructura con todos sus campos, la envía mediante el comando `send()`. El envío de esta información implica el envío de dos mensajes diferentes:

- Un número entero correspondiente al tamaño de la estructura, es decir, la cantidad de datos que va a recibir el servidor a continuación. Esto es así, dado que existen dos tipos de ficheros distintos, y en el caso del fichero aleatorio, los atributos carretera, detector, sentido y fecha estarán vacíos.
- La estructura con los atributos: Se enviará la estructura de datos teniendo en cuenta el tamaño de la misma.

El siguiente paso es que el servidor reciba los datos correctamente. Para ello debe disponer en el código de dos `recv()`, correspondientes a los dos envíos que realiza el cliente:

- En el primer `recv()`, el servidor debe saber que va a recibir un entero.
- En el segundo `recv()`, el servidor debe tener en cuenta que va a recibir la estructura del mensaje cuyo tamaño se corresponderá con los datos recibidos en la recepción inmediatamente anterior.

Posteriormente, lo que hace el servidor es enviar una confirmación al cliente de que se han introducido los datos correctamente en la base de datos. El funcionamiento es el mismo, el servidor debe usar el comando `send()` y el cliente el comando `recv()`.

5.4. Diseño de la base de datos

El diseño de una base de datos consiste en definir la estructura de los datos que debe tener la base de datos de un sistema de información [16]. En la Figura 48, se puede observar un ejemplo de diseño de la base de datos pasando por las tres etapas que se explican a continuación.

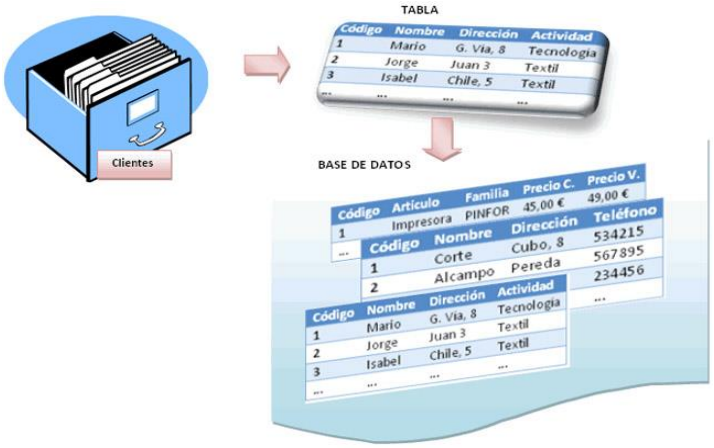

Figura 48: Ejemplo de diseño de base de datos

El diseño de la base de datos se va a dividir en tres etapas:

- Etapa del diseño conceptual: Es donde se va a desarrollar el modelo entidad relación (ER). La base de datos se va a diseñar teniendo en cuenta los dos posibles ficheros que pueden ser gestionados. Como se verá en el siguiente capítulo la diferencia entre el fichero aleatorio y el obtenido gracias a la DGT es que en el segundo aparecen las variables: carretera, detectores, sentido y fecha y en el otro no.

SENSOR
carretera (VARCHAR (3)) detector (VARCHAR (25)) sentido (VARCHAR (3)) fecha (VARCHAR (18)) intensidad (INT) ocupación (INT) velocidad (INT) longmedia (INT) distmedia (INT) agrupme7 (INT) agrupma7 (INT) velm50 (INT) velme100 (INT) velma100 (INT)

Tabla 26: Modelo E/R

- Etapa del diseño lógico: Se tiene en cuenta el diseño anterior para posteriormente adaptarlo a la tecnología usada. La base de datos MySQL va a poseer catorce columnas, las cuales corresponden a los datos que va a introducir el servidor cuando se conecte.

- Etapa del diseño físico: En esta etapa se parte del resultado anterior para que se adapte a la tecnología a usar para obtener una mayor eficiencia. No se puede mejorar nada ya que con el diseño lógico se adapta totalmente a la tecnología, por lo que no se ha utilizado.

6. Implementación

Una vez que conocemos las funciones que se van a desarrollar en nuestro sistema (análisis) y que se han decidido los distintos componentes y cómo han de implementarse (diseño) es el momento de empezar a programar.

6.1. Entorno de desarrollo

En este apartado se va a comentar las herramientas de trabajo que se han utilizado para desarrollar el TFG. Para desarrollar la aplicación del control del tráfico de vehículos es necesario tener un entorno de desarrollo de Linux, bien teniendo el Sistema Operativo Linux como sistema operativo nativo o bien como máquina virtual.

6.1.1. Máquina Virtual

Una máquina virtual es un software que simula una computadora y puede ejecutar varios programas como si fuese una computadora real [20]. Existen varios tipos de máquina virtuales, las dos más conocidas son: VirtualBox y VMware. Ambas han sido utilizadas en distintos ordenadores para poder simular el sistema operativo Linux, que es el que se asume para el desarrollo del sistema.

Figura 49: Logo de VirtualBox

Figura 50: Logo de VMware

6.1.2. Linux Ubuntu

Ubuntu es un sistema operativo completo basado en GNU/Linux, disponible de forma libre con soporte para la comunidad y los profesionales, que permite realizar todas las tareas diarias que actualmente llevas a cabo con tu máquina sin necesidad de reaprender todo lo que ya sabes [23].

Está desarrollado por una gran comunidad mundial en la que todas las personas que quieran colaborar para mejorarlo son bienvenidas, beneficiándose y beneficiando a los demás con sus aportes, opiniones, ideas y participación activa en general.

Ubuntu GNU/Linux es un sistema libre porque cumple con las siguientes características que definen toda pieza de software libre [23]:

- La libertad de usar el programa, con cualquier propósito.
- La libertad de estudiar cómo funciona el programa y modificarlo, adaptándolo a tus necesidades.
- La libertad de distribuir copias del programa, con lo cual puedes ayudar a tu prójimo.
- La libertad de mejorar el programa y hacer públicas esas mejoras a los demás, de modo que toda la comunidad se beneficie.

Figura 51: Logo de Ubuntu

6.2. Ejecución del cliente y del servidor

Como se vio en el capítulo 2, un socket es una puerta de comunicación entre el proceso de aplicación del cliente y del servidor. Para la realización de la programación con sockets TCP el cliente debe contactar con el servidor, para ellos se deben cumplir varios requisitos [24]:

- El servidor debe estar ejecutándose primero.

```
./server -p 1200
```

  Se observa cómo se arranca el servidor en el puerto 1200 *(*opción *-p 1200).*
- El cliente por su parte, debe ejecutar posteriormente y especificar en línea de comandos los siguientes argumentos:
  - -s dirección IP. la dirección IP del servidor y
  - –p puerto: puerto del servidor
  - –r Origen de los datos: si va a leer el fichero de la DGT (opción –r 0); por el contrario si genera un fichero aleatorio (opción –r 1).

Se observa que el cliente es ejecutado apuntando hacia el servidor que se encuentra en la misma máquina (-s *localhost),* en el puerto 1200 *(-p 1200)* y fichero de la DGT *(-r 0).*

```
./cliente -s localhost -p 1200 -r 0
```

Se observa que el cliente es ejecutado apuntando hacia el servidor que se encuentra en la misma máquina (-s *localhost),* en el puerto 1200 *(-p 1200)* y creando un fichero aleatorio *(-r 1).*

```
./cliente -s localhost -p 1200 -r 1
```

- Una vez se ejecutan el cliente y el servidor se comienza estableciendo una conexión entre ambos. En cada conexión que un cliente realiza contra el servidor, el servidor crea otro socket para poder comunicarse con cada cliente simultáneamente. Esta característica le otorga al servidor el carácter de concurrente.

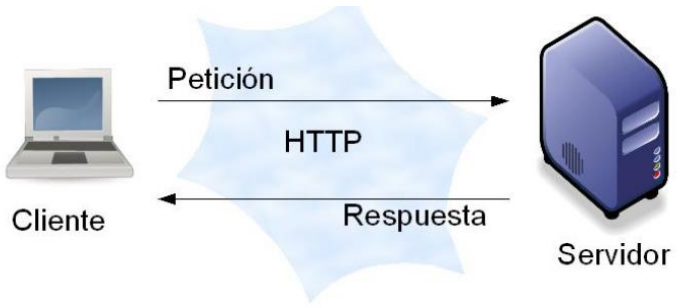

Figura 52: Comunicación sockets entre cliente y servidor

6.2.1. Implementación del servidor

El servidor es una parte fundamental de la aplicación que va a recibir la información de los clientes y va a gestionar la base de datos. Por ello en este apartado se va a explicar las partes básicas de su implementación.

6.2.1.1. Librerías

Las librerías son una colección de clases y funciones, escritas en el núcleo del lenguaje. La biblioteca estándar proporciona varios contenedores genéricos, funciones para utilizar y manipular esos contenedores, funciones objeto, cadenas y flujos genéricos.

Las librerías utilizadas para el código del servidor son las siguientes:

```
#include <stdio.h>
#include <stdlib.h>/* Funciones de ficheros */
#include <strings.h>
#include <string.h>
#include <unistd.h>
#include <sysexits.h>
#include <ctype.h>
#include <wordexp.h>
#include <sys/time.h>//
```

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <sys/stat.h> /* Para la estructura stat */
#include <fcntl.h>      /* Modos de apertura */
#include <mysql/mysql.h>
#include <pthread.h>/*hilos*
```

6.2.1.2. Socket del servidor

Para la creación del socket del servidor los pasos que se han seguido son los siguientes:

En primer lugar se declara el descriptor del socket, esta es una variable que se saca fuera del main, por lo tanto es una *variable global.*

```
int sd;
```

También se declara otro descriptor que va a ser el que se reciba del cliente.

```
int sc;
```

Una vez declarados, lo que se hace es abrir el socket del servidor, se declara el protocolo de comunicación que este caso es TCP y la estructura la información de la familia, la dirección y el puerto.

```
if ((sd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP))<0)
           printf ("SERVER: Error en el socket");
val = 1;
setsockopt(sd, SOL_SOCKET, SO_REUSEADDR, (char *) &val,
sizeof(int));

bzero((char *)&server_addr, sizeof(server_addr));
server_addr.sin_family= AF_INET;
server_addr.sin_addr.s_addr = INADDR_ANY;
server_addr.sin_port = htons(port);
```

Una vez hecho esto, se realiza el `bind()`, comprobando que se ha realizado correctamente y en caso contrario se sale del programa.

```
if(bind(sd, (struct sockaddr *)&server_addr, sizeof(struct
sockaddr_in))==-1){
      perror("bind");
      exit(1);
}
```

El siguiente paso es realizar el `listen()` cuyo objetivo es preparar como pasivo el socket del servidor, es decir, habilitarlo para que pueda aceptar conexiones entrantes.

```
if(listen(sd, 5)==-1){
      perror(" listen\n");
      exit(1);
}
```

Dentro de un bucle infinito (ya que el servidor únicamente para cuando se realiza en consola "*Ctrl+C*") se ejecuta `accept()`, cuya utilidad es aceptar la conexión entrante de un cliente. Esta operación crea un nuevo socket que será usado para las posteriores comunicaciones entre el servidor y ese cliente.

```
sc = accept(sd, (struct sockaddr *)&cliente_addr,&size);
            if(sc==-1){
                  printf("Error al enviar");
                  exit(1);
            }
```

6.2.1.3. Conexión a la base de datos

Otra de las partes fundamentales de la implementación del servidor es la conexión a la base de datos. Para realizarlo, se hace necesaria la declaración de tres variables fundamentales que permitan definir el nombre del servidor, el nombre del usuario y la contraseña del usuario:

```
#define HOST "localhost"    //   nombre   del   servidor   local   o
direccion del servidor 127.0.0.1
#define USER "root"         // nombre del usuario

#define PASS ""             // password

#define DB "regla"          // nombre de la base de datos

MYSQL *con;                 //  variable  de  conexión  para  una  DB
MySQL

MYSQL_RES *res;             //  variable que contendrá el resultado
de la consuta

MYSQL_ROW row;              //  variable  que  contendrá  los  campos
por cada registro consultado
```

Una vez declaradas estas variables el siguiente paso es realizar la conexión con la base de datos:

```
con=mysql_init(NULL);
if  (!mysql_real_connect(con,  HOST,  USER,  PASS,  DB,  MYSQL_PORT,
NULL, 0)){
            fprintf(stderr, "%s\n", mysql_error(con)); /* si hay un
error definir cual fue dicho error */
            mysql_close(con);
            exit(1);
      }
```

A continuación se crean los métodos que se van a utilizar para que el servidor establezca una conexión con la base de datos. Los métodos que realizan operaciones sobre la base de datos son los siguientes:

- agrega: Este método como su propio nombre indica inserta los datos que llegan del cliente al servidor a la base de datos.

```
agrega (con,Sensor.carretera,Sensor.detector, Sensor.sentido,
Sensor.fecha,Sensor.intensidad,
Sensor.ocupacion,Sensor.velocidad, Sensor.longmedia,
Sensor.distmedia,Sensor.agrupm7, Sensor.agrupM7,
Sensor.velm50,Sensor.velm100,Sensor.velM100);
```

- muestra: Este método muestra todos los datos que contiene la tabla *sensor* de la base de datos. Esta tabla está compuesta por 14 columnas, las cuales coinciden con los atributos que se envían en la estructura.

```
muestra (con, "SELECT * FROM sensor;", row, res);
```

6.2.1.4. Servidor concurrente

La implementación del servidor concurrente se ha hecho a través de procesos ligeros o hilos. Cada vez que llega una conexión entrante se crea un nuevo hilo el cual se encargará de introducir los datos de la estructura que llega a la base de datos. Esto se realizará en el método `trata_hilos()`.

```
pthread_create( &hilo, &attr1, (void*)trata_hilos, &sc);
```

El método creado lo que realiza es:

- Recibe el número de líneas que va a recibir del cliente (sensor)
- Espera que llegue toda la información.
- Accede a la base de datos para agregar la información recibida.
- Muestra la situación de la base de datos por consola.

```
int n;
n=recv (s_local, &linea, sizeof (int),0);
if(n==-1){
      printf("Error al leer \n");
      exit(1);
}
int prueba=0;
printf("Numero de linea %d \n", linea);

while(prueba<linea){

      printf("Linea %d \n", prueba);
      prueba++;
      if((recv (s_local, &Sensor, sizeof (struct sensor),0))<0){;
// recibe la petición
            printf("Error al leer: \n");
            exit(1);
      }
      printf("Intensidad: %d \n ", Sensor.intensidad);

      pthread_mutex_lock(&mymutex);
      agrega (con,Sensor.carretera,Sensor.detector, Sensor.sentido,
Sensor.fecha,Sensor.intensidad, Sensor.ocupacion,Sensor.velocidad,
Sensor.longmedia, Sensor.distmedia,Sensor.agrupm7, Sensor.agrupM7,
Sensor.velm50,Sensor.velm100,Sensor.velM100);
      fprintf (stdout, "\nDatos actuales en la DB\n");
      muestra (con, "SELECT * FROM reglas;", row, res);
```

```
        pthread_mutex_unlock(&mymutex);
}
```

6.2.1.5. Compilación y ejecución

Para compilar el servidor es necesario tener en cuenta las librerías que se han utilizado, el comando para compilar el servidor es el siguiente:

```
gcc -L/usr/lib64/mysql server.c -o server -lm -lmysqlclient -lpthread
```

Para ejecutar el servidor, simplemente hay que indicar el puerto por el que está activo, como se muestra a continuación, a través del puerto 12000:

```
./server -p 12000
```

6.3. Implementación del cliente (sensor)

Los clientes o sensores, que en esta aplicación son sinónimos, son los que van a proporcionar la información al servidor del estado de la carretera simulando a lo que harían los dispositivos en una mini-ciudad inteligente.

A continuación se va a explicar cómo se ha implementado el cliente.

6.3.1. Librerías

Las librerías utilizadas para el código del cliente son las siguientes:

```
#include <stdio.h>
#include <stdlib.h>/* Funciones de ficheros */
#include <strings.h>
#include <string.h>
#include <unistd.h>
#include <sysexits.h>
#include <ctype.h>
#include <wordexp.h>
#include <sys/time.h>//
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <sys/stat.h> /* Para la estructura stat */
#include <fcntl.h>      /* Modos de apertura */
#include <time.h>
```

6.3.2. Socket del cliente

La implementación del socket del cliente se ha realizado de la siguiente forma:

En primer lugar se declara el descriptor del socket, esta es una variable que se saca fuera del main, por lo tanto es una *variable global*.

```
int sd;
```

Una vez declarado, lo que se realiza es abrir el socket del cliente, se declara el protocolo de comunicación que este caso es TCP y la estructura la información de la familia, la dirección y el puerto.

```
sd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
bzero((char *)&server_addr, sizeof(server_addr));
server_addr.sin_family= AF_INET;
hp = gethostbyname (server);
memcpy (&(server_addr.sin_addr), hp->h_addr, hp->h_length);
server_addr.sin_port = htons(port);
memset (&(server_addr.sin_zero),'\0',8);
```

El siguiente paso es conectar el socket con el servidor mediante `connect()` y comprobar que no dé error al realizar la conexión.

```
int c;
c=connect(sd, (struct sockaddr *) &server_addr,
sizeof(server_addr));
if(c==-1){
      printf("Error en la conexión con el servidor: %s, en el
puerto: %s\n", server, port_s);
      exit(1);
}
```

En el cliente no es necesario hacer `bind()` porque el propio sistema operativo se encargará de hacerlo. El cliente se conecta con el servidor que debe encontrarse bloqueado en una llamada `accept()`.

Una vez conectados servidor y cliente, el cliente enviará el número de líneas que posee el fichero que le va a entregar a través del comando `send()` y se comprueba que no dé error el envío.

```
if(send (sd, &num_lineas, sizeof(int),0) < 0){
      printf("Error al enviar lineas");
      exit(1);
}
```

Posteriormente lo que hace el cliente es leer del fichero, ya sea del fichero de la DGT o del generado aleatoriamente e irá leyendo línea a línea el fichero, rellenando la estructura con los datos y enviando la estructura al servidor.

```
if(send(sd, &Sensor, sizeof(struct sensor), 0) < 0){
      printf("Error al enviar \n");
      exit(1);
}
```

6.3.3. Generación del fichero aleatorio

La generación del fichero aleatorio se realiza a través de una función llamada *rdtsc()*, la cual posee una semilla que va asociada al número de ciclos utilizados por el procesador desde el inicio. Esta función sólo es válida para procesadores x86 (Intel, AMD, etc.).

```
int rdtsc()
{
     __asm__ __volatile__("rdtsc");
}
```

A continuación, se van a explicar los *thresholds* o umbrales utilizados para cada uno de los parámetros que envían los sensores:

- Intensidad: El threshold usado para la intensidad es de 20.
- Ocupación: El threshold usado para la ocupación es de 100.
- Velocidad: El threshold usado para la velocidad es de 120.
- Longitud media: El threshold usado para la longitud media es de 175.
- Distancia media: El threshold usado para la distancia media es de 350.
- Agrupación de vehículos menores de 7 metros: El threshold usado para la agrupación es el número aleatorio correspondiente a la intensidad.
- Agrupación de vehículos mayores de 7 metros: El threshold usado para la agrupación es el número aleatorio correspondiente a la intensidad menos la agrupación de vehículos menores de 7.
- Velocidad de vehículos que van entre 50 y 100 es el número aleatorio correspondiente a la intensidad.
- Velocidad de vehículos que van a más velocidad de 100: El threshold usado es el número aleatorio correspondiente a la intensidad menos los que vehículos que van entre 50 y 100.
- Velocidad de vehículos que van a menos de 50: El threshold usado es el número aleatorio correspondiente a la intensidad menos los vehículos que van entre 50 y 100 y los vehículos que van a más de 100.

Es importante tener en cuenta que en el caso de que la intensidad sea 0, el resto de variables también van a serlo.

6.3.4. Compilación y ejecución

Para compilar el cliente o sensor no es necesario introducir un comando adicional ya que no utiliza librerías adicionales. El comando para compilar el cliente es:

```
gcc cliente.c -o cliente
```

Para ejecutar el cliente, hay que tener en cuenta la IP a la que se va a realizar la conexión, el puerto de conexión y si se realiza a través del fichero de la DGT o el fichero generado aleatoriamente.

6.4. Implementación de la base de datos

En este apartado se describirá la implementación realizada para la base de datos MySQL. La implementación de la base de datos se basa en dos partes:

- Creación de la base de datos.
- Creación de las tablas y atributos.

6.4.1. Creación de la base de datos

Lo primero que se va a realizar para poder crear correctamente la base de datos es comprobar qué bases de datos están creadas de tal manera que así no se va a repetir el nombre [27].

A través del comando `SHOW DATABASES` se verán qué bases de datos están creadas.

Figura 53: Mostrar bases de datos

A continuación, se crea la base de datos con el nombre elegido, en este caso va a ser "*sensor*".

Figura 54: Crear base de datos

Posteriormente, volvemos a utilizar el comando `SHOW DATABASES` y se comprueba que la base de datos se ha creado correctamente:

Figura 55: Mostrar bases de datos instaladas

Para finalizar, usamos el comando `USE` para comenzar a utilizar la nueva base de datos creada.

Figura 56: Seleccionar base de datos

6.4.2. Creación de las tablas y atributos

Una vez creada la base de datos es necesario crear correctamente las tablas junto con sus atributos. Para esta aplicación simplemente es necesaria una tabla con varios atributos, para crearla se utiliza el siguiente comando:

```
CREATE TABLE prueba (carretera VARCHAR (3), detector VARCHAR (25),sentido VARCHAR(3), fecha VARCHAR(18), intensidad INT, ocupacion INT, velocidad INT, longmedia INT, distmedia INT, agrupme7 INT, agrupma7 INT, velm50 INT, velme100 INT, velma100 INT);
```

6.5. Reglas *Smart Traffic*

En este último apartado se explica cómo se han implementado las reglas que posee la "*carretera inteligente*". Para que las reglas funcionen es necesario que los puntos anteriores estén implementados correctamente, desde la conexión cliente-servidor, al hecho de guardar los datos en la base de datos.

Para la ejecución de las reglas, el servidor accederá a la base de datos. Dado que el cometido principal del servidor es aceptar conexiones de los clientes en el programa servidor se creará un nuevo hilo cuyo cometido será la ejecución de las reglas un periodo dado. En esta carretera inteligente de nuestra *Smart City* se van a desarrollar tres reglas independientes, cuyo objetivo es hacer que los ciudadanos mejoren su vida diaria y más concretamente, sepan en qué estado se encuentra la carretera a la que ellos van a acceder. Estas reglas son:

- Regla 1: Regla de mejora de velocidad del tráfico.
- Regla 2: Regla de la distancia de seguridad.
- Regla 3: Regla de la ocupación.

6.5.1. Regla 1: Regla de mejora de velocidad del tráfico

Todas las carreteras poseen una velocidad máxima. Los datos que envían los sensores contienen una velocidad media a la que circulan los vehículos en la carretera. Por lo tanto, dependiendo del valor de la velocidad media, se pueden extraer conclusiones acerca de la densidad del tráfico de la carretera. Para ilustrarlo, usaremos un código de colores que permita identificar fácilmente el estado de la carretera. Los colores que usamos son:

- Rojo: Indica que la velocidad media de los vehículos es inferior al 25% de la velocidad máxima permitida en la vía.
- Amarillo: Indica que la velocidad media de los vehículos es superior al 25% pero inferior al 75% de la velocidad máxima permitida en la vía.
- Verde: Indica que la velocidad media de los vehículos es superior al 75% de la velocidad máxima permitida en la vía.

Para la implementación de la regla 1, se han de seguir los siguientes pasos:

- Crear un fichero donde se almacenarán los valores de velocidad consultados de la base de datos. Para poder ofrecer los datos de la carretera a los usuarios de la vía y llevar un control de las detecciones realizadas, es necesario utilizar un fichero que permita almacenar en soporte no volátil esta información,

```
if((ficheroSalida = fopen("regla1.txt", "wt"))==NULL){
     printf ( " Error en la apertura. Es posible que el
fichero no exista \n "); //Control del error de apertura
     exit(-1);
}
```

- Conectar con la base de datos. Como todos los datos se guardan en la base de datos, es necesario conectarse a ella para obtenerlos, por lo que se comprueba que la conexión es correcta y que la aplicación puede continuar.

```
if (mysql_query(con, consulta)==0){
           res= mysql_use_result(con);
}
```

- Obtener los datos de la velocidad guardados. Una vez que se ha conectado a la base de datos, se accede al atributo `velocidad` y se escribe en el fichero. Mediante una variable auxiliar, se va a contar el número de datos que posee dicha columna.

```
     while ((row=mysql_fetch_row(res))){
                fputs(row[6], ficheroSalida);
```

```
                fputs("\n", ficheroSalida);
                cont++;
            }
```

- Calcular la media de la velocidad de los vehículos. Con los datos en el fichero, se accede al mismo para hacer los cálculos matemáticos, es decir, la media de las velocidades de los vehículos.

```
while ((ch = fgetc(entrada)) != EOF){
      auxInt[cor]= ch;
      cor++;
      if (ch == '\n'){
            cor--;
            //printf("CONTADOR COR: %d \n", cor);
            if(cor<4){
                  if(cor==3){
                        aux=   (auxInt[0]-48)*100+   (auxInt[1]-
48)*10+ auxInt[2]-48;
                        //printf("aux: %d \n", aux);
                  }
                  if (cor==2){
                        aux= (auxInt[0]-48)*10+(auxInt[1]-48);
                        //printf("aux: %d \n", aux);
                  }
                  if(cor==1){
                        aux= auxInt[0]-48;
                        //printf("aux: %d \n", aux);
                  }
                  vel+=aux;
                  //printf("Velocidad %d \n", vel);
            }
            cor=0;

      }
}
```

- Escribir en fichero los datos obtenidos. Una vez realizados los cálculos, se escribe en el fichero la velocidad media y el color que posee el tramo.

    - En caso de que el tramo esté en verde.

```
      if (media>(0.75*VEL_MAX_VIA)){
      fprintf(entrada, "\nEl tramo está en color verde porque
la velocidad media de los vehículos es mayor que el 75%% de
la velocidad de la carretera. \n\n");
}
```

    - En caso de que el tramo esté en amarillo.

```
if((0.75*VEL_MAX_VIA)>media && media>(0.25*VEL_MAX_VIA)){
      fprintf(entrada, "El tramo está en color amarillo porque
la velocidad media de los vehículos es menor que el 75%% de
la velocidad máxima permitida y mayor que el 25%% \n\n");
}
```

    - En caso de que el tramo esté en rojo.

```
if(media<(VEL_MAX_VIA*0.25)){
      fprintf(entrada, "El tramo está en color rojo porque la
velocidad media de los vehículos es menor que el 25%% de la
velocidad máxima permitida. \n\n");
}
```

6.5.2. Regla 2: Regla de la distancia de seguridad

Otro de los datos obtenidos de los sensores es la distancia que existe entre dos vehículos consecutivos, para detectar que se cumple la distancia de seguridad e intentar evitar que éstos estén excesivamente cerca. Una de las posibles acciones a desarrolla es que si se detecta que la distancia entre vehículos es inferior a la permitida, se podrían mostrar en las pantallas de aviso de la carretera un mensaje avisando a los conductores que deben respetar la distancia de seguridad.

El *threshold* que se utiliza en esta ocasión para la distancia de seguridad es el valor de 180. Si la distancia media de los vehículos es menor de 180 se considera que no mantienen la distancia de seguridad y viceversa.

Para la implementación de la regla 2, se han de seguir los siguientes pasos:

- Los primeros cuatros pasos son equivalentes a los ejecutados en la regla número 1.

- Escribir en fichero los datos obtenidos. Cuando se han calculado los datos, se escribe en el fichero si la distancia media de seguridad entre los coches es la adecuada, es decir, si es segura.

```
if (media<(180)){
      fprintf(entrada, "\nAumente la distancia de seguridad.
\n");
}
else{
      fprintf(entrada, "\nDistancia de seguridad media es
correcta porque supera los 180m establecidos. \n");
}
fclose(entrada);
```

6.5.3. Regla 3: Regla de la ocupación

Como ya se he definido anteriormente, la ocupación es el porcentaje de tiempo que la espira está ocupada. Este dato es otro parámetro que ofrecen al servidor los sensores y da otra perspectiva de cómo se encuentra la carretera.

Para poder detectar el estado de ocupación se realizará la media de los datos de ocupación y se comparará con un determinado *threshold*. Por ello se puede subdividir en tres niveles de ocupación: alta, media o baja.

- Ocupación baja: Ocurre cuando el porcentaje medio de ocupación de la carretera es inferior al 25%.
- Ocupación media: Ocurre cuando el porcentaje medio de ocupación de la carretera es superior al 25% pero inferior del 75%.
- Ocupación alta: Ocurre cuando el porcentaje medio de ocupación de la carretera es superior al 75%.

Para la implementación de la regla 3, se han de seguir los siguientes pasos:

- Los primeros cuatros pasos coinciden con la regla número 1 y 2.

- Escribir en fichero los datos obtenidos. Cuando se han calculado los datos, se escribe en el fichero la ocupación media de los datos guardados en la base de datos.

```
if (media>(75)){
      fprintf(entrada, "\nLa ocupación media supera el 75%%.
\n");
}
if((75)>media && media>(25)){
      fprintf(entrada, "La ocupación media de los vehículos es
menor que el 75%% y mayor que el 25%%. \n");
}
if(media<(25)){
      fprintf(entrada, "La ocupación media de los vehículos es
menor que el 25%%. \n");
}
fclose(entrada);
}
```

7. Evaluación

En este capítulo se van a realizar las pruebas de evaluación del sistema y pruebas encaminadas a evaluar su rendimiento tomando distintos escenarios. Asimismo, se desarrollará una evaluación de los datos del fichero de la DGT.

7.1. Pruebas de evaluación del sistema

En este apartado, se va a mostrar que todos los requisitos especificados en el análisis han sido cumplidos. Para cada requisito se muestra la prueba realizada y el resultado obtenido.

- **F1: Disponibilidad del servidor:**

El servidor debe estar siempre escuchando conexiones entrantes de los clientes. Esta validación se demuestra con el código del servidor, el cual se encuentra en un bucle infinito y nunca va a parar de escuchar salvo que manualmente se pare la ejecución mediante la señal Ctrl+C o un error inesperado en la máquina donde ejecute el servidor.

```
while(1){
      //Escuchar a los clientes que lo soliciten
}
```

- **F2: Puerto del servidor**

El servidor debe escoger el puerto por el cual se van a aceptar las conexiones, ese puerto debe encontrarse entre el 1024 y el 65335. Como se puede observar en la Figura 57, si no se encuentra entre ese intervalo el servidor no va a ejecutarse y la aplicación no funcionará.

```
administrador@ubuntu:~/Escritorio/29-05$ ./server -p 1023
Error: Port must be in the range 1024 <= port <= 65535
Usage: ./server [-d] -p <port>
administrador@ubuntu:~/Escritorio/29-05$
```

Figura 57: Conexión servidor puerto fallido

En la Figura 58, se observa que si se usa un puerto del intervalo correctamente el servidor se ejecutará y la aplicación funcionará correctamente.

```
administrador@ubuntu:~/Escritorio/29-05$ ./server -p 1200
-> Conectado a la Base de datos: regla
```

Figura 58: Conexión servidor puerto correcto

- **F3: Servidor concurrente**

El servidor debe ser capaz de recibir más de una petición a la vez. Por ejemplo, si n clientes intentan conectar al servidor, se crearán n procesos ligeros independientes. Para

proteger el acceso a la base de datos de posibles condiciones de carrera, se protegerá la inserción de los para que pueda realizarse en exclusividad.

A continuación, en la Figura 59 se muestra cómo el servidor acepta que se ejecuten dos clientes a la vez.

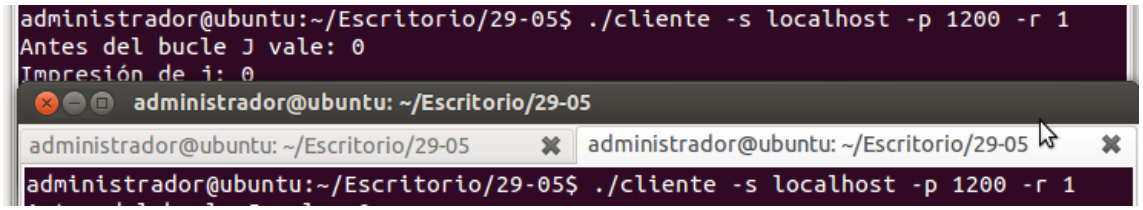

Figura 59: Ejecución de dos cliente simultáneamente

- **F4: Almacenar los datos que reciben los sensores**

El servidor debe almacenar los datos que recibe de los sensores en la base de datos de manera ordenada, tal y como se describe en el diseño (véase capítulo 5). En la Figura 60 se muestra un ejemplo de cómo se almacenan los datos y cómo se organizan según van llegando.

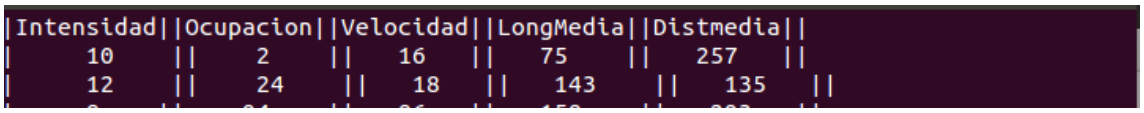

Figura 60: Organización de la base de datos

- **F5: Calcular y visualizar la velocidad media de los vehículos**

Este requisito demuestra que la regla 1, explicada en el capítulo 6, funciona correctamente. Para demostrarlo, en la Figura 61 se muestra la escritura en el fichero de salida, donde aparecen los datos de la media de velocidad calculados y el color en el que se encuentra la carretera.

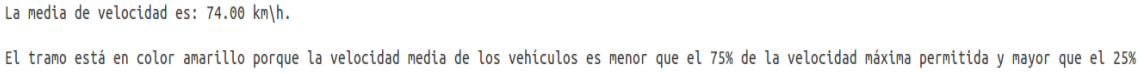

Figura 61: Ejecución de la regla 1

- **F6: Calcular y visualizar la distancia media de los vehículos**

Este requisito demuestra que la regla 2, explicada en el capítulo 6, funciona correctamente. Para demostrarlo, en la Figura 62 se muestra el fichero donde aparecen los datos de la distancia media calculados.

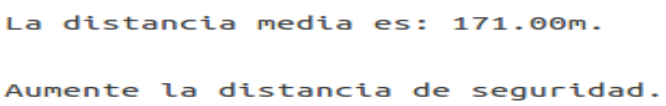

Figura 62: Ejecución de la regla 2

- **F7: Calcular y visualizar la ocupación media de los vehículos**

Este requisito demuestra que la regla 3, explicada en el capítulo 6, funciona correctamente. Para demostrarlo, en la Figura 63 se muestra el fichero donde aparecen los datos de la ocupación media.

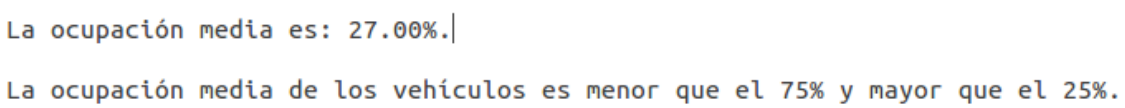

Figura 63: Ejecución de la regla 3

- **F8: Ejecución de las reglas de tráfico periodo establecida**

En el diseño de la aplicación, se estableció que un proceso se ejecutara en un periodo determinado para ejecutar las reglas 1, 2 y 4. Cada vez que entra en ejecución dicho proceso se calcularan las tres reglas establecidas. En la Figura 64, se muestra cómo el proceso ejecuta las reglas que posteriormente mostraran en los ficheros lo que se ve en los tres requisitos anteriores.

Figura 64: Ejecutando reglas

- **F9: Mostrar el estado del servidor**

El servidor debe mostrar su estado para facilitar a los clientes (sensores) el hecho de si se pueden conectar o no. Como se muestra en la Figura 65, el servidor muestra que está disponible, más concretamente, “Esperando conexión”.

Figura 65: Estado del servidor

- **F10: El cliente debe elegir el puerto correcto**

La aplicación entre el cliente y el servidor debe estar conectada a través del mismo puerto, por lo que si el cliente se intenta conectar al servidor por un puerto por el que no está disponible, la aplicación no funcionará. Como se muestra en la Figura 66.

```
jaime@JaimePerruni:~/Escritorio/27-05$ ./cliente -s localhost -p 1201 -r 1
Error en la conexión con el servidor: localhost, en el puerto: 1201
```

Figura 66: Error de conexión de puertos

En el caso de que sí se conecte por el puerto adecuado, la conexión se realizará correctamente. Figura 67.

```
jaime@JaimePerruni:~/Escritorio/27-05$ ./cliente -s localhost -p 1200 -r 1
```

Figura 67: Correcta conexión cliente servidor

- **F11: Envío de los datos de los clientes al servidor**

El cliente debe enviar los datos al servidor a partir del fichero origen de los datos, ya sea el proporcionado por la DGT o el fichero aleatorio. Se ha comprobado que si en el envío de los datos da algún error, el cliente sale del programa ya que está gestionado así en el código.

- **F12: Generar datos aleatorios para simulaciones largas**

Como se ha explicado a lo largo de toda la memoria, el cliente debe generar un fichero con datos aleatorios para simular el comportamiento de los sensores. A continuación en la Figura 68 se muestra un ejemplo del fichero generado.

```
18;50;85;21;181;3;15;0;17;1
3;54;11;64;234;2;1;3;0;0
16;6;3;4;209;8;8;0;14;2
```

Figura 68: Ejemplo de fichero aleatorio generado

- **F13: Cliente manda datos leídos del fichero de la DGT**

El cliente debe mandar los datos del fichero de la DGT, en el caso de que haya sido escogida esa opción, por lo que hay que controlar el caso de que dé error en la lectura, para ello se tiene el siguiente código que lo comprueba.

- **F14: Sensor envía datos de la carretera**

Para que este requisito se cumpla hay que tener en cuenta el número de columnas que posee la base de datos y cómo está organizado el fichero que se va a leer. A continuación la Figura 69 muestra un ejemplo de fichero que debe leer el sensor.

```
A-6;DET12 PK011+400D T01 A-6;DEC;01/11/2013  00:01;8;2;100;38;124;7;0;0;3;4
A-6;DET12 PK011+400D T01 A-6;DEC;01/11/2013  00:02;5;1;94;41;179;5;0;0;4;1
A-6;DET12 PK011+400D T01 A-6;DEC;01/11/2013  00:03;3;1;106;42;196;3;0;0;1;2
```

Figura 69: Ejemplo de fichero DGT a leer

- **F15: Creación de la base de datos**

Este requisito se ha validado creando la base de datos *reglas* y observando el resultado que se imprime en la consola MySQL:

```
mysql> create database reglas;
Query OK, 1 row affected (0.00 sec)

mysql>
```

Figura 70: Creación de la base de datos

- **F16: Crear la tabla**

En el requisito F16, se especifica la creación de la tabla que almacene los datos del sensor. Por esa razón se deben crear los 14 atributos ya comentados anteriormente:

```
mysql> CREATE TABLE sensor (carretera VARCHAR (3), detector VARCHAR (25),sentido
 VARCHAR(3), fecha VARCHAR(18), intensidad INT, ocupacion INT, velocidad INT, lo
ngmedia INT, distmedia INT, agrupme7 INT, agrupma7 INT, velm50 INT, velme100 INT
, velma100 INT);
Query OK, 0 rows affected (0.06 sec)
```

Figura 71: Creación tabla en base de datos

- **F17: Sensores deben leer datos fichero DGT**

Para leer el fichero de la DGT es necesario abrirlo y comprobar que al hacerlo no ha dado ningún error, por lo que lo primero que se debe hacer es comprobar que el fichero se abre correctamente.

Una vez comprobado que no da error, el cliente comienza a leer el fichero hasta que llega el final.

- **F18: Servidor debe mostrar los datos a los usuarios**

Como se puede observar en la Figura 61, 62 y 63 que corresponden a los requisitos F5, F6 y F7, el servidor muestra correctamente los datos obtenidos de la base de datos a los usuarios.

- **F19: Ubicación del fichero**

La ubicación del fichero debe estar en la misma carpeta que donde se encuentran los ficheros con el código ejecutable del programa. En el caso de que no estén, el cliente no podrá acceder al fichero de la DGT para leerlo y el servidor expondrá los resultados de la base de datos en la carpeta en la que se encuentre.

Como se muestra en la Figura 72, todos los ficheros necesarios se encuentran en la misma carpeta.

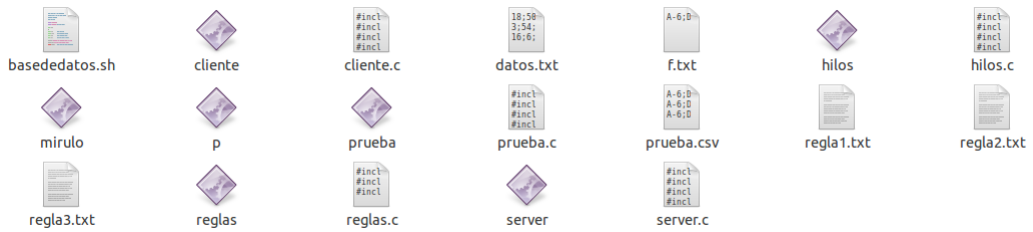

Figura 72: Ubicación correcta del fichero

- **F20: Atributos entre “;”**

Para que el cliente pueda leer los ficheros los atributos deben separarse mediante el carácter “;”; si esta condición no se cumple no se podría separar correctamente los atributos por cada columna. Por tanto, el contenido de los ficheros debe respetar esta condición. . Las siguientes Figuras 73 (fichero de la DGT) y 74 (figura aleatorio) Muestran sendos ejemplos del contenido de los ficheros.

```
A-6;DET12 PK011+400D T01 A-6;DEC;01/11/2013  00:01;8;2;100;38;124;7;0;0;3;4
A-6;DET12 PK011+400D T01 A-6;DEC;01/11/2013  00:02;5;1;94;41;179;5;0;0;4;1
A-6;DET12 PK011+400D T01 A-6;DEC;01/11/2013  00:03;3;1;106;42;196;3;0;0;1;2
```

Figura 73: Ejemplo de fichero DGT con ;

```
18;50;85;21;181;3;15;0;17;1
3;54;11;64;234;2;1;3;0;0
16;6;3;4;209;8;8;0;14;2
```

Figura 74: Ejemplo de fichero aleatorio con ;

- **F21: SSOO Linux tanto en cliente como servidor**

El Sistema Operativo donde se encuentra la aplicación debe ser Linux o encontrarse en una máquina virtual que posea un sistema Linux también.

- **F22: Lenguaje de programación C en servidor**

Para que la aplicación funcione, el lenguaje de programación utilizado en el servidor debe ser lenguaje C, como se encuentra en la Figura 75.

server.c ×

```
int n;
n=recv (s_local, &linea, sizeof (int),0);
if(n==-1){
        printf("Error al leer \n");
        exit(1);
}
int prueba=0;
printf("Numero de linea %d \n", linea);
```

Figura 75: Lenguaje C para servidor

- **F23: Lenguaje de programación C en cliente**

Para que la aplicación funcione, el lenguaje de programación utilizado en el servidor debe ser lenguaje C, como se encuentra en la Figura 76.

cliente.c ×

```
#include <stdio.h>
#include <stdlib.h>/* Funciones de ficheros */
#include <strings.h>
#include <string.h>
#include <unistd.h>
#include <sysexits.h>
#include <ctype.h>
```

Figura 76: Lenguaje C para cliente

- **F24: Lenguaje de programación MySQL en base de datos**

La base de datos debe ser MySQL, como se muestra en la Figura 77.

```
CREATE DATABASE prueba;
```

Figura 77: Lenguaje MySQL para base de datos

- **F25: Uso de sockets**

Como se ha explicado en el capítulo 5, el de diseño, los sockets son necesarios para la comunicación entre cliente y servidor, sino, la aplicación no se podría realizar.

7.2. Pruebas de rendimiento

En este apartado se van a evaluar aspectos de rendimiento de la aplicación. En particular, se va a medir el tiempo que tarda el servidor en recibir los datos en dos escenarios: cuando el cliente y servidor residen en la misma máquina (local) y cuando residen en distintas máquinas (remoto).

7.2.1. Escenario 1: local

Las pruebas de rendimiento en local se realizan conectando el servidor y el cliente dentro de una misma máquina. Por ello, intuitivamente, sabemos que el tiempo de ejecución de las pruebas va a ser menor que trabajando en remoto como se verá en el siguiente apartado.

A continuación se muestra una gráfica del tiempo que tarda el cliente en enviar los datos al servidor. Para hacer una estimación media más exacta, se han tomado 15 muestras de tiempo y partir de ahí se obtienen los cálculos medios.

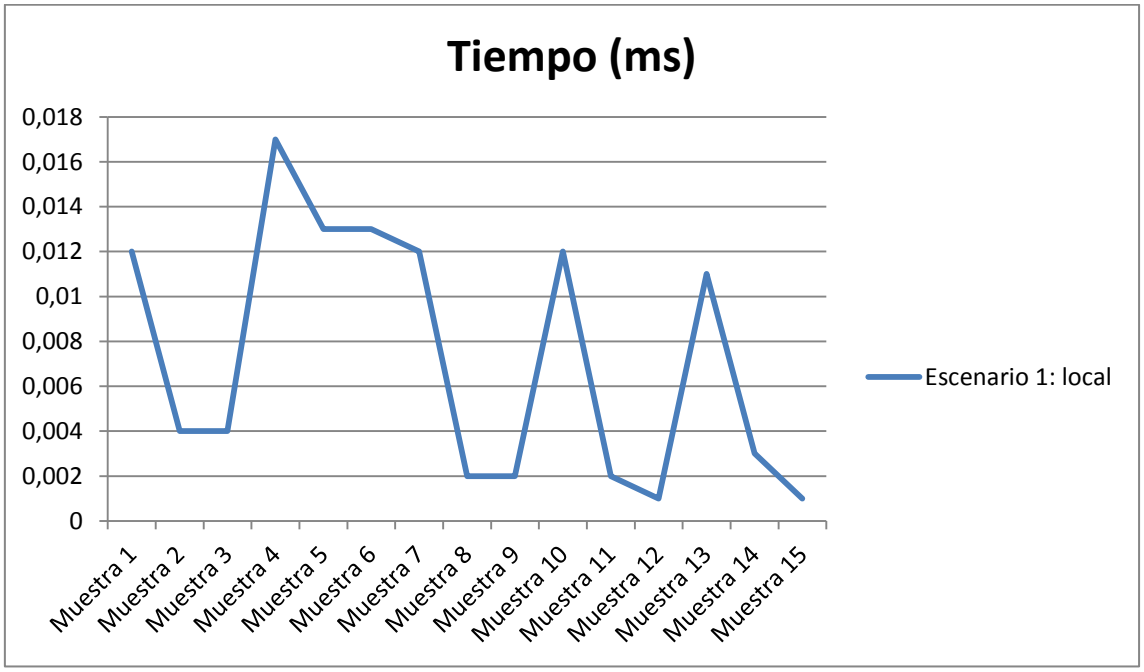

Figura 78: Prueba de rendimiento en local

La media de tiempo de envío en local se calcula de la siguiente forma:

$$\bar{X} = \frac{\sum_{i=1}^{n} x_i}{n}$$

X= (0.012+0.004+0.004+0.017+0.013+0.013+0.012+0.002+0.001+0.011+0.003+0.001) /15= 0.0062 milisegundos.

7.2.2. Escenario 2: remoto

Las pruebas de rendimiento en remoto se realizan conectando el servidor y el cliente en máquinas distintas, por lo que el tiempo de ejecución va a ser mayor que trabajando en remoto.

En la Figura 79, se muestran los resultados obtenidos de 15 muestras. Como se puede apreciar los valores son mayores que los mostrados en la Figura 78.

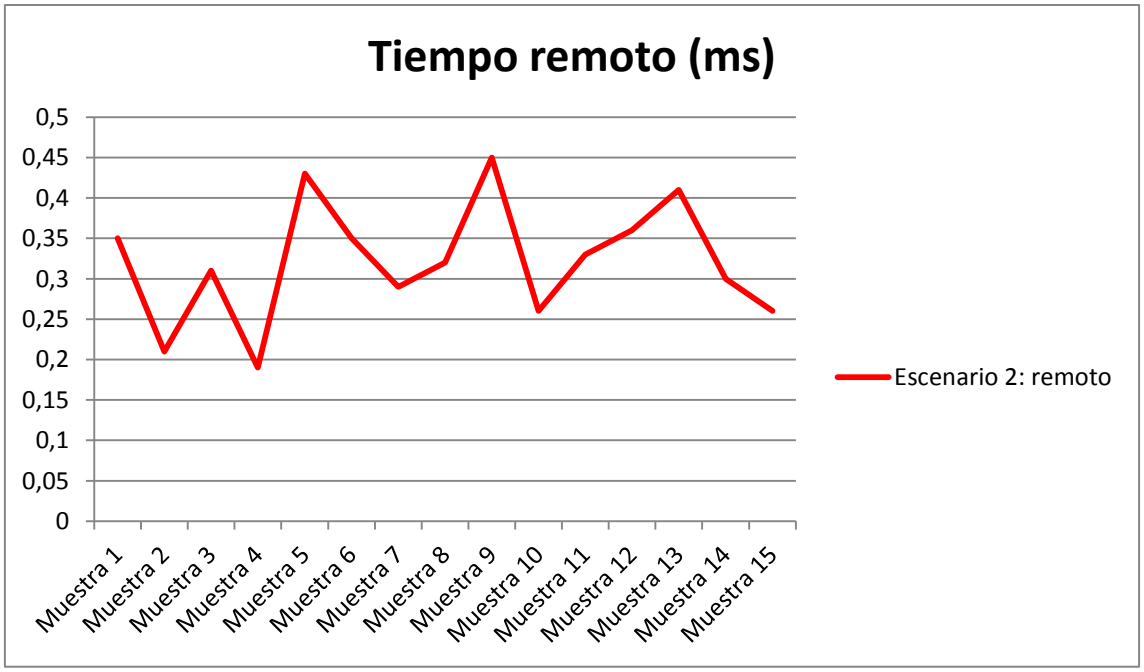

Figura 79: Prueba de rendimiento en remoto

La media de tiempo de envío en remoto se calcula de la siguiente forma:

$$\bar{X} = \frac{\sum_{i=1}^{n} x_i}{n}$$

X= (0.35+0.21+.019+0.43+0.35+0.29+0.32+0.45+0.26+0.33+0.36+0.41+0.3+0.26)/15 = 0.3213 milisegundos.

7.2.3. Comparación entre local y remoto

En la Figura 80, se muestra una comparación de tiempos para cada muestra en remoto y en local. Teniendo en cuenta los valores medios calculados anteriormente, se puede razonar sobre la proporción en que es mayor el tiempo remoto frente al tiempo local, mediante:

$$\frac{Media\ de\ remoto}{Media\ local}$$

El resultado de esta fórmula es. 0.3213/0.0062= 51.8279, por lo que tarda más de 50 veces enviar los datos en remoto que en local.

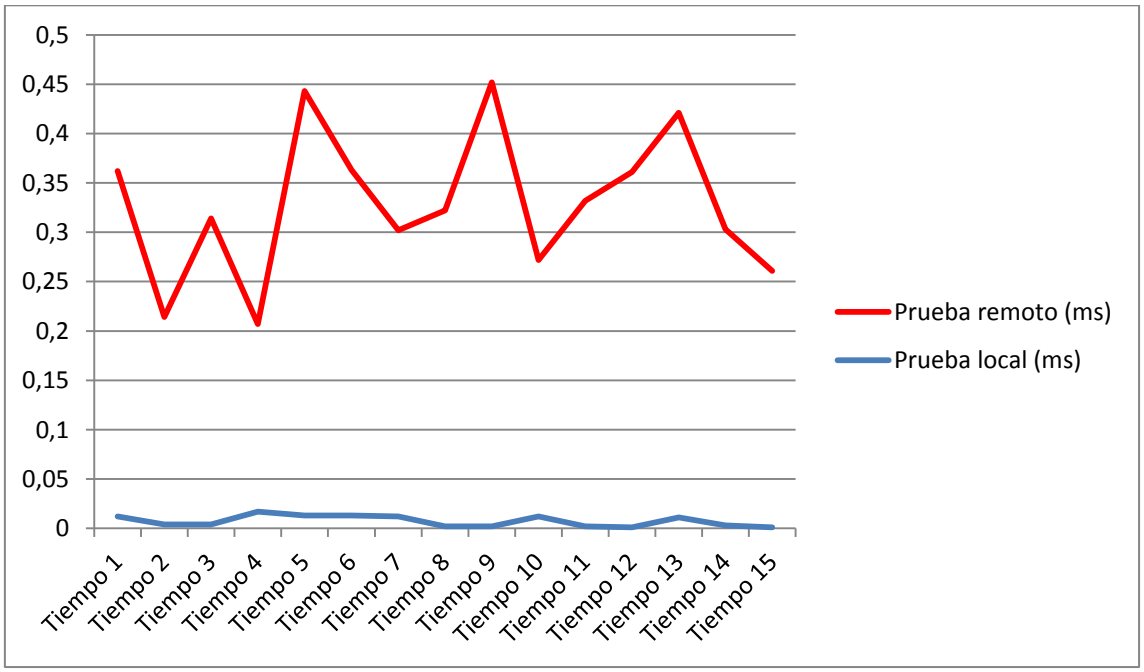

Figura 80: Comparación entre local y remoto

7.3. Evaluación de las mejoras gracias a la tecnología de la *Smart city*

En este apartado, se van a explicar las posibles mejoras que añade la *Smart city* para con los vehículos, es decir, las facilidades que da a los usuarios mostrando información de la carretera, desde su ocupación, intensidad y velocidad.

Según los estudios realizados y gracias a las muestras de información a los usuarios, éstos deciden si ir por la carretera por la que iban a pasar o cambiar de ruta para no quedarse atrapados en un atasco, suponiendo en este caso que la ocupación sea superior al 75%. De esta manera, los vehículos no van por esta carretera y contribuyen a mejorar las características principales estudiadas en las reglas.

Estos datos son extraídos de porcentajes estimados y no representan fielmente la información que puede llegar a haber en la carretera pero sí reflejan la posible mejora que tendría la aplicación en la sociedad actual.

Partiendo del apartado de Ingeniería del tráfico del capítulo 2, se estima que la intensidad es directamente proporcional a la ocupación y a la velocidad de los vehículos en la carretera, por lo que a partir de ahí se pueden obtener los datos que se pueden llegar a mejorar con la aplicación.

Después de hacer el estudio de cómo puede influir las *Smart cities* en la circulación de los vehículos, se puede llegar a mejorar un 20% la intensidad de la carretera. Este porcentaje incluido es un porcentaje supuesto ya que no se disponen de los medio suficientes para probarlo.

La Tabla 30 muestra cómo puede influir el hecho de que los vehículos tengas conocimiento del estado en el que se encuentra la carretera.

Ocupación (%)	Velocidad (km/h.)	Datos teóricos	Datos mejorados
2	100	2	3
1	94	1	1
1	106	1	1
2	93	2	2
2	104	2	3
1	101	1	1
1	90	1	1
1	92	1	1
0	0	0	0
2	99	2	2
2	98	2	2
1	98	1	1

Tabla 27: Datos teóricos frente a mejorados

Los datos que aparecen en la tabla son los siguientes:

- Ocupación: Es el porcentaje de tiempo que la espira que obtiene los datos de los vehículos está ocupada.
- Velocidad: Es la velocidad media de los vehículos en un tiempo determinado.
- Datos teóricos: La columna de datos teóricos corresponde a la intensidad calculada a través de la fórmula del capítulo de ingeniería del tráfico. Es decir,

  Datos teóricos= (Ocupación * Velocidad)/100
- Datos mejorados: La columna de datos mejorados corresponde a las posibles mejoras que se puede llegar a tener gracias a la muestra de la información de la situación de la carretera.

  Datos mejorados= (Ocupación * Velocidad)/80

Por lo que se puede observar en la Tabla 30, en la columna de "*Datos teóricos"* sólo aparecen los vehículos que actualmente hay en la carretera, teniendo en cuenta que la información la ven otros vehículos, aparece la columna de "*Datos mejorados*", la cual demuestra cómo el tráfico mejora gracias a la información ofrecida por el servidor.

8. Conclusiones

En este capítulo se van a repasar los objetivos fijados en el capítulo 1 para determinar si han sido cumplidos o no y poder evaluar si la consecución del proyecto ha sido satisfactoria. Este capítulo presenta también los trabajos futuros que se podrían llegar a realizar a partir de este trabajo. Por último se presenta el presupuesto y un conjunto de conclusiones sobre este TFG.

8.1. Evaluación de los objetivos

Al comienzo del proyecto, en el capítulo 1.2 Objetivos, se definieron los objetivos que se pretendían realizar con la finalización del mismo. A continuación, y con el proyecto ya finalizado, se va a estudiar cada objetivo y se va a detallar el estado en el que se encuentra cada uno.

- Estudio de los conceptos de IoT y en particular de *Smart Cities*. Dado el alto impacto que puede tener en el futuro de nuestras ciudades, ha sido necesario revisar la bibliografía relacionada con estos dos conceptos.

  Al comenzar el proyecto, es necesario entender los conceptos de los que se va a hablar a lo largo de toda la aplicación. De hecho sin buscar información acerca de ellos y estudiarlo bien, el proyecto no podría haberse llevado a cabo, por lo que se puede afirmar después de toda la documentación encontrada que el objetivo se ha cumplido.

- Simulación de una carretera inteligente en el contexto de una *Smart City*. Dicha carretera inteligente permitirá monitorizar mediante sensores el estado de una carretera. Estos datos sobre el estado serán enviados en tiempo real a un servidor externo el cuál, almacenará, procesará y determinará situaciones anómalas, además de aportar alguna solución para las mismas.

  La aplicación desarrolla una mini ciudad inteligente con su funcionamiento básico que es el envío y procesamiento de información gracias a sensores y un servidor, por lo que la simulación es correcta y en consecuencia el objetivo está completado.

- Análisis, diseño e implementación de una aplicación cliente y servidor que permita implementar la simulación anterior. En particular, se deberá:

  - Creación de un servidor concurrente que sea capaz de gestionar las peticiones de los clientes.

    Se ha creado un servidor programado en C que es capaz de estar esperando constantemente las peticiones de los clientes y gestionarlas posteriormente, además se tiene en cuenta el hecho de que puedan llegar varias peticiones a la vez por lo que se puede afirmar que se ha creado un servidor concurrente.

- Creación de sensores (clientes) que detecten información de la carretera y la envíen al servidor.

  Los sensores o clientes son capaces de detectar la información correctamente, una vez almacenada en el mensaje la envían al servidor.

- Creación de una base de datos donde se guardará toda la información que haya sido enviada por lo clientes a través del servidor que es quien conecta con esta.

  Se ha creado una base de datos en MySQL donde se guarda toda la información que llega al servidor desde los sensores.

- Creación de un protocolo de comunicación que explote las funcionalidades del servidor:

  Este objetivo es uno de los más importantes del proyecto, sin él la aplicación no funcionaría ya que los sensores no podrían enviar los datos y éstos se perderían cuando se hiciese la siguiente detección. El protocolo de comunicación elegida es TCP y se ha realizado mediante sockets-

- Facilitar información a los usuarios del estado en el que se encuentra la carretera.

  Una de las finalidades de IoT, o más concretamente de las Smart Cities es facilitar la vida diaria a los ciudadanos, por ello este objetivo consiste en facilitar los datos que se han almacenado en la base de datos a los usuario para que ellos vean en qué estado se encuentra la vía y si quiere utilizarla o no.

- Asegurar la fiabilidad del sistema.

  Actualmente el sistema es fiable, ya que no presenta ningún tipo de fallo, pero sí es cierto que se podría mejorar la fiabilidad haciendo el sistema más complejo. El siguiente apartado de líneas futuras comenta cómo hacer el sistema más fiable.

8.2. Trabajos futuros

El desarrollo de este proyecto ha dejado algunas líneas abiertas. A pesar de que la aplicación cliente-servidor es totalmente funcional y cumple con los objetivos del proyecto se pueden realizar mejoras funcionales que la hagan más eficiente, funcional y segura. Por ello, a continuación se explicarán los posibles trabajos futuros y con ello las mejoras que se pueden realizar:

- Compresión de los datos: Los datos almacenados en la base de datos no están comprimidos. El límite que posee la base de datos es la capacidad del disco duro de la CPU. Por ello, se podrían guardar los datos en lugar de cada minuto,

guardarlos cada 15 minutos, de tal forma que en una misma base de datos se podría guardar 15 veces más datos.

- Tolerancia a fallos: Se podría especificar algún método para que en el caso de que el servidor diese un fallo o los propios sensores, se pudiese reparar el sistema rápidamente sin provocar que algún dato sea guardado.
- Seguridad del sistema: El sistema actualmente no tiene seguridad por lo que esta mejora se podría dividir en dos líneas futuras:
    - Cifrado: La base de datos está en claro por lo que cualquier amenaza podría atacar el sistema y obtener los datos que le interesen, no sólo eso, podría modificar los datos haciendo que los usuarios obtuvieran datos incorrectos de la realidad. Con el cifrado se lograría la integridad del sistema.
    - Administrador: La creación de un administrador exclusivo para la base de datos haría que sólo éste podría controlar la base de datos, por lo que ninguna otra persona podría acceder a los datos. Con la creación del administrador los datos estarían autenticados y existiría una segregación de tareas, que forma parte de los principios de seguridad de un sistema.
- Interfaz de usuario: Para mejorar la aplicación creada, se podría crear una interfaz que mejorara la usabilidad de cara al usuario, por ejemplo que no fuera necesario que el usuario tuviera que ejecutar la aplicación con un comando en consola, simplemente que tuviera una interfaz que lo hiciera por sí solo.
- Mejorar fiabilidad del sistema (autenticación, clientes erróneos o falsos): Otra de las posibles mejoras de la aplicación es la autenticación de los clientes y del servidor entre ellos a través de un protocolo como puede ser SSL, ya que un sensor podría enviar datos erróneos para modificar las medias reales. Un servidor sin validar también podría recibir información de los sensores y podría usar esos datos para fines no benéficos.

8.3. Presupuesto

Los presupuestos de proyectos muestran cuánto va a costar completarlo. Éstos usualmente detallan los gastos necesarios, estos gastos se van a dividir en tres grandes grupos: recursos humanos, recursos materiales y servicios subcontratados.

A continuación se va a desglosar lo que supone cada uno de ellos.

8.3.1. Recursos humanos

El coste humano se puede establecer mediante la siguiente tabla:

Persona	Fase	Precio/hora (€)	Horas	Dedicación (Hombre/mes)	Coste de la fase (€)
Analista	Análisis del entorno	32	60	1.6	1.920
Diseñador	Diseño	35	96	2.1	3.360
Ingeniero	Implementación	30	175	2.75	5.250
Ingeniero	Evaluación y pruebas	30	83	1.55	2.490
Documentador	Documentación	10	65	1	650
	Total		479	9	13.670

Tabla 28: Presupuesto recursos humanos

8.3.2. Recursos materiales

Los recursos materiales se pueden apreciar en la siguiente tabla:

Elemento	Cantidad	Coste (€)	Subtotal (€)
Ordenador de sobremesa	1	750	750
Portátil	1	400	400
Total			1.150
Total amortizado			172,50

Tabla 29: Presupuesto recursos materiales

El total amortizado se ha calculado mediante la siguiente fórmula:

(A/B)*C*D

A=número de meses de la fecha de facturación en el que el equipo es utilizado

B= periodo de depreciación (60 meses)

C= coste del equipo (sin IVA)

D= % del uso que se dedica al proyecto (habitualmente 100%)

8.3.3. Servicios subcontratados

En la siguiente tabla se pueden apreciar los servicios subcontratados para la realización del proyecto.

Elemento	Cantidad	Coste (€)	Subtotal (€)
Conexión a internet	1	980	980
Impresión documentos	1	50	50
Total			1.030

Tabla 30: Presupuesto servicios subcontratados

8.3.4. Costes totales

En la siguiente tabla se muestran los costes totales de todo el proyecto:

Concepto	Coste (€)
Recursos humanos	13.670
Amortización de recursos humanos	173
Servicios subcontratados	1.030
Costes indirectos	2.975
Total	17.847

Tabla 31: Presupuesto costes totales

8.3.5. Plantilla reunión

En la siguiente plantilla se muestran todos los presupuestos unidos:

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor:
Jaime Morales Rodríguez de Lope
2.- Departamento:
Departamento de informática
3.- Descripción del Proyecto:

- Título	Diseño, implementación y evaluación de una aplicación de control del tráfico de vehículos
- Duración (meses)	9
Tasa de costes Indirectos:	20%

4.- Presupuesto total del Proyecto (valores en Euros):
Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a título informativo)	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)	Firma de conformidad
Jaime Morales Rodríguez de Lope		Analista	1,6	1.200,00	1.920,00	
Jaime Morales Rodríguez de Lope		Diseñador	2,1	1.600,00	3.350,00	
Jaime Morales Rodríguez de Lope		Ingeniero	4,3	1.800,00	7.740,00	
Jaime Morales Rodríguez de Lope		Documentación	1	650,00	650,00	
					0,00	
		Hombres mes 9		Total	13.670,00	

[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
Ordenador de sobremesa	750,00	100	9	60	112,50
Portátil	400,00	100	9	60	50,00
					0,00
					0,00
					0,00
					0,00
				Total	172,50

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste cel equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
Impresión documentos	Copistería	50,00
	Total	50,00

OTROS COSTES DIRECTOS DEL PROYECTO[e]

Descripción	Empresa	Costes imputable
Servicio de Internet	Telefónica	980,00
	Total	980,00

[e] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas,

6.- Resumen de costes

Presupuesto Costes Totales	Presupuesto Costes Totales
Personal	13.670
Amortización	173
Subcontratación de tareas	50
Costes de funcionamiento	980
Costes Indirectos	2.975
Total	17.847

8.4. Valoración personal

Después de tanto tiempo deseando que llegara este momento, por fin ha llegado, el final de la carrera. Este Trabajo de Fin de Grado demuestra todos los conocimientos que he ido adquiriendo a lo largo de la carrera.

No todo lo aprendido es con respecto a mis estudios de Ingeniería Informática, de hecho estos cinco años me han ayudado a aprender cosas importantes, a ser mejor persona, madurar y sobre todo a aprender lo que más vale en la vida: el esfuerzo y la dedicación.

Ese esfuerzo y dedicación es lo que me ha llevado hasta aquí, el hecho de estar trabajando y estudiando requiere de un mayor esfuerzo pero va directamente relacionado con la satisfacción obtenida al ver que después de tanto, lo he conseguido acabar.

Para finalizar, quería agradecer a todos los profesores y compañeros que he tenido a lo largo de la carrera que me han ayudado a llegar hasta aquí.

8.5. Colaboraciones

Quiero agradecer a las siguientes personas el hecho de que me prestaran su tiempo y colaboraran en que este TFG fuera posible:

José Luis Chica Moreu	Licenciado en Ciencias e Ingeniero Técnico de Obras Públicas	Director del Centro de Gestión de Tráfico de Madrid Dirección General de Tráfico	DGT Dirección General de Tráfico
José Tomás Hernández	Ingeniero Superior de Telecomunicaciones	Director Técnico de Accesos a Madrid (UTE) Indra Sistemas	indra
Rafael Tato Sánchez	Ingeniero Técnico Industrial	Director Accesos a Madrid (UTE) Indra Sistemas	indra
Rafael Morales Morales	Teniente Coronel de la Guardia Civil	Jefe del Sector de Tráfico de Madrid Agrupación de Tráfico Dirección General de la Guardia Civil	GUARDIA CIVIL TRÁFICO

Tabla 32: Agradecimientos

9. Bibliografía

[1] IERC_Cluster_Book_2012_WEB

[2] http://www.cisco.com/web/ES/assets/executives/pdf/Internet_of_Things_IoT_IBSG_0411FINAL.pdf

[3] http://www.cisco.com/web/ES/assets/executives/pdf/Internet_of_Things_IoT_IBSG_0411FINAL.pdf IoT

[4] http://blogs.salleurl.edu/networking-and-internet-technologies/el-reto-de-internet-of-things/

[5] http://www.libelium.com/es/top_50_iot_sensor_applications_ranking/

[6] http://smartcity-telefonica.com/?p=65

[7] http://www.numerex.com/global-m2m-spanish

[8] http://smartcity-telefonica.com/page-flip/informe_anual.pdf

[9] OEP 2013 Especialidad: Gestión Técnica del Tráfico Elaborado en 2011. Ingeniería del tráfico.

[10] http://ocw.uoc.edu/computer-science-technology-and-multimedia/bases-de-datos/bases-de-datos/P06_M2109_02151.pdf

[11] https://www.mindomo.com/mindmap/ventajas-y-desventajas-de-la-base-de-datos-my-sql-402f5022676f47919f2a15e79534330b

[12] http://mysqldaniel.wordpress.com/ventajas-y-desventajas/

[13] http://www.iot-spain.com/?lang=es

[14] http://wwwhatsnew.com/2013/11/23/como-funciona-google-maps/ http://blogs.salleurl.edu/networking-and-internet-technologies/el-reto-de-internet-of-things/ IoT

[15] http://www.ecured.cu/index.php/Lenguaje_de_Programaci%C3%B3n_C

[16] http://ocw.uoc.edu/computer-science-technology-and-multimedia/bases-de-datos/bases-de-datos/P06_M2109_02150.pdf

[17] Accenture_FTF_Internet_de_las_Cosas_2011

[18] Datos_A-6_pk_11+400D_1minuto_20131101

[19] http://es.wikipedia.org/wiki/IBM

[20] http://es.wikipedia.org/wiki/M%C3%A1quina_virtual

[21] http://es.wikipedia.org/wiki/VirtualBox

[22] http://www.alegsa.com.ar/Dic/vmware.php

[23] http://ubuntu.org.ar/?q=node/2

[24] https://www.tlm.unavarra.es/~daniel/docencia/rc_itig/rc_itig04_05/slides/clase9y10-SocketsTCP.pdf

[25] http://cbasesdedatos.blogspot.com.es/

[26] http://es.kioskea.net/faq/2818-generar-numeros-aleatorios-eficazmente-con-rand

[27] Mysql_en_c_sEc_undersecurity.pdf

[28] http://profecarolinaquinodoz.com/principal/?tag=concepto-de-redes-clienteservidor

[29] http://es.wikipedia.org/wiki/Direcci%C3%B3n_General_de_Tr%C3%A1fico

[30] http://www.angelfire.com/trek/storwald/Sockets.pdf

[31] http://es.tldp.org/Universitarios/seminario-2-sockets.html

[32] http://es.wikipedia.org/wiki/Red_de_sensores